Générateurs de nombres véritablement aléatoires à base d'anneaux asynchrones : conception, caractérisation et sécurisation

Abdelkarim Cherkaoui

Thèse présentée pour obtenir le grade de docteur
de l'université Jean Monnet de Saint-Etienne

N° bibliothèque : 978-2-11-129193-5

École doctorale Sciences, Ingénierie et Santé

Spécialité : Micro-électronique

Générateurs de nombres véritablement aléatoires à base d'anneaux asynchrones

Conception, caractérisation et sécurisation

Par : **Abdelkarim Cherkaoui**

Thèse soutenue publiquement le **16 Juin 2014** devant le jury composé de :

Président :	**Lionel TORRES**, Professeur, Université de Montpellier II
Rapporteurs :	**Sylvain GUILLEY**, Professeur, TELECOM ParisTech
	Christian PIGUET, Professeur, Ecole Polytechnique Fédérale de Lausanne (EPFL)
Examinateurs :	**Nele MENTENS**, Professeur assistant, Université catholique de Leuven (KU Leuven)
	Yannick TEGLIA, Docteur, Expert en sécurité, STMicroelectronics
	Emmanuel PROUFF, HDR, Expert en sécurité, Agence Nationale de Sécurité des Systèmes d'Information (ANSSI)
	Viktor FISCHER, Professeur, Université Jean Monnet de Saint-Etienne, **directeur de thèse**
	Laurent FESQUET, Maître de conférences, Institut National Polytechnique de Grenoble (INPG), **co-directeur de thèse**
Membres invités :	**Alain AUBERT**, Maître de conférences, Université Jean Monnet de Saint-Etienne, **co-encadrant**

Thèse préparée au
Laboratoire Hubert Curien (UMR CNRS 5516)
18 Rue Benoit Lauras
42 000, SAINT-ETIENNE
En collaboration avec le laboratoire
TIMA (UMR CNRS 5159)
46 Avenue Félix Viallet
38 000, GRENOBLE
Et financée par
La région Rhône-Alpes

A mes parents,

Sommaire

Introduction générale

Cette thèse traite de générateurs de nombres véritablement aléatoires exploitant le bruit électronique dans des structures oscillantes appelées anneaux auto-séquencés. Nous présentons une architecture matérielle permettant de générer des nombres aléatoires destinés à être utilisés dans les applications cryptographiques exigeant un haut niveau de sécurité ainsi qu'une méthodologie complète pour la dimensionner, la caractériser, et la durcir d'un point de vue sécurité. Ces travaux s'inscrivent principalement dans le domaine de la cryptographie appliquée. Les disciplines abordées regroupent la conception de circuits numériques (en particulier la conception d'oscillateurs asynchrones) ainsi que l'analyse et modélisation statistique.

Contexte scientifique

Les questions de cyber-sécurité et de sécurité matérielle sont plus que jamais d'actualité. Ces trois dernières années ont été particulièrement denses en faits d'attaques informatiques de grande ampleur (*Areva* en 2011, Iran en 2012, Corée du Sud en 2013), scandale d'espionnage à grande échelle (révélations de Snowden sur les programmes d'écoute de la NSA en 2013) et mise en évidence de failles de sécurité importantes dans des dispositifs extrêmement sensibles (faille de sécurité dans des FPGA embarqués dans des engins de combat américains en 2012 [SW12], attaque sur un système électronique de frein ABS présentée à la conférence CHES 2013 [SMTS13]). Ces derniers mois, il semble du plus en plus avéré que du matériel cryptographique critique contient des failles introduites intentionnellement pour des fins d'espionnage [1] ou des buts commerciaux (renseignements sur les habitudes des clients, publicité). Le matériel électronique grand public est largement touché : il y a peu de temps, des experts découvraient un réseau d'objets connectés, parmi lesquels figurait un réfrigérateur, qui envoyaient des spams à l'insu de leurs utilisateurs. Les attaques visant à intercepter des communications cryptées sont également de plus en plus ciblées, et exploitent souvent des failles présentes dans le matériel électronique. Par ailleurs, l'usage intensif des communications sans-fil (via les smartphones, tablettes tactiles et autres appareils communicants) rend leur interception plus facile. La demande en sécurité (au niveau logiciel et matériel) est donc de plus en plus importante et touche aussi bien le grand public que les états, ceux-ci déploient des budgets de plus en plus conséquents pour faire face aux menaces liées à la sécurité de l'information.

La cryptographie est l'art de chiffrer, *i.e.* de rendre inintelligible un message à autre que qui-de-droit. C'est une science qui date des débuts de la civilisation ; dès l'antiquité, elle était enseignée et tenue en haute estime par les écoles tactiques de la Grèce, les généraux de l'armée romaine et les

1. En Septembre 2013, *RSA Security* conseille à ses clients de ne plus utiliser un générateur de nombres pseudo-aléatoire destiné aux applications cryptographiques (Dual_EC_DRBG) qui a été certifié par le NIST américain http ://arstechnica.com/security/2013/09/stop-using-nsa-influence-code-in-our-product-rsa-tells-customers/

armées des empires chinois et perses. Dans les récits des guerres du premier empire, il est souvent question de communications cryptées : on distribuait aux généraux deux clés pour correspondre entre eux et avec l'état-major général. Dès la fin du $19^{ième}$ siècle, la correspondance par clés chiffrées était adoptée par la plupart des armées d'Europe. Par la suite, la cryptographie moderne a connu son essor au milieu du $20^{ième}$ siècle, s'appuyant sur les bases théoriques posées par le mathématicien américain Claude Shannon en 1949, qui est aujourd'hui considéré comme le père fondateur de la théorie de l'information. Outre le domaine militaire, les réseaux informatiques et les télécommunications au sens large du terme, les services offerts par la cryptographie touchent à beaucoup d'aspects de la vie quotidienne : les transactions bancaires, les cartes à puces (cartes SIM, sécurité sociale, cartes bancaires ...), le matériel médical, les appareils de divertissement, etc.

La cryptographie, selon sa définition moderne, étudie tous les mécanismes destinés à garantir la sécurité de l'information échangée pendant une communication : sa confidentialité (l'information ne doit être accessible qu'à ceux dont l'accès est autorisé), son authenticité (l'identité de l'utilisateur doit être vérifiée pour lui autoriser ou lui interdire l'accès) et son intégrité (garantir l'intégralité, la précision, l'authenticité et la validité des données lors de la transmission). La plupart des algorithmes cryptographiques modernes s'appuient sur le principe bien connu de Kerckhoffs (cryptologue néerlandais du $19^{ième}$ siècle) : la sécurité d'un système de chiffrement doit reposer uniquement sur le secret de la clé de chiffrement, l'algorithme de chiffrement étant connu par tous les partis. Ainsi, pour attaquer de tels systèmes, il faut théoriquement tester toutes les clés (attaque par force brute). Celles-ci sont générées de manière automatisée, et elles sont de plus en plus grandes afin de faire face à l'efficacité de ces attaques et à l'augmentation fulgurante de la puissance de calcul : les attaquants peuvent avoir des budgets conséquents (par exemple un état) ou ils peuvent mobiliser des moyens très importants (réseau d'ordinateur infectés).

Comme la génération des clés cryptographiques est automatisée, il est important que celles-ci soient imprévisibles et qu'elles aient de bonne propriétés statistiques afin de minimiser toute fuite d'information qui permettrait d'accélérer une attaque par force brute et de réduire son coût. Par exemple, un attaquant peut exploiter la connaissance de propriétés d'un sous-ensemble de clés (comme le biais, la périodicité, la présence de dépendances ...) pour améliorer son attaque en réduisant l'ensemble des hypothèses à tester. Ou pire encore, il pourrait prédire une séquence de clés avec une probabilité non négligeable (exemple de clés ayant de bonnes propriétés statistiques, mais générées selon un schéma algorithmique partiellement ou totalement connu). Les clés de chiffrement sont donc générées par des modules dédiés, appelés générateurs de nombres aléatoires (*Random Number Generators - RNG*).

Il n'est pas possible de générer des nombres réellement aléatoires de manière algorithmique : au mieux, on peut générer des nombres qui ont de très bonnes propriétés statistiques mais qui restent théoriquement prévisibles si l'algorithme est connu [2]. De tels générateurs sont appelés générateurs de nombres pseudo-aléatoires (*Deterministic Random Number Generators - DRNG* [3]), ce sont des constructions mathématiques (ils peuvent être implantés dans le matériel et le logiciel). Par contre,

2. John von Neumann déclare en 1951 dans *Various techniques used in connection with random digits* : "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin"

3. Bien que l'expression *Deterministic Random* nous semble contradictoire, DRNG est un sigle souvent utilisé dans la littérature scientifique pour désigner ces générateurs

il est possible d'extraire des nombres véritablement aléatoires à partir d'un phénomène physique imprévisible (phénomènes quantiques, bruit électronique ...). Dans ce cas, si le mécanisme d'extraction est correctement mis en place, les nombres générés peuvent avoir de bonnes propriétés statistiques, mais surtout ils peuvent être imprévisibles. De tels générateurs sont appelés générateurs de nombres véritablement aléatoires (*True Random Number Generators - TRNG*[4]), ce sont des constructions mécaniques ou électriques (implantées dans le matériel). Les DRNG modernes ont l'avantage d'être rapides, faciles à implanter et de générer systématiquement des nombres avec de bonnes propriétés statistiques. Ils peuvent parfois être utilisés dans des applications cryptographiques, mais il est préférable que leur algorithme utilise une graine d'initialisation issue d'un TRNG (pour le critère d'imprévisibilité). Celle-ci doit, de plus, être renouvelée périodiquement en fonction de la périodicité de l'algorithme. Les TRNG sont donc nécessaires dans tout système de chiffrement contemporain embarqué dans le matériel si le niveau de sécurité visé est haut. Ils sont également un bloc extrêmement sensible d'un point de vue sécurité (principe de Kerckhoffs), et doivent être conçus, caractérisés et évalués rigoureusement.

Problématique

La conception et l'évaluation de la qualité et des performances des TRNG posent problème dans la mesure où elles ne se font pas sur des critères simples comme la consommation, le débit ou la taille. Bien sûr la qualité statistique est validée à l'aide de batteries de tests standard mais ce n'est pas un critère éliminatoire dans le cas des TRNG : les défauts statistiques peuvent être corrigés à l'aide d'algorithmes de post-traitement adaptés. Ainsi, dans les applications cryptographiques (notamment la génération de clés de chiffrement), il est préférable de disposer de séquences imprévisibles mais contenant des défauts statistiques (qui peuvent être corrigés) que des séquences parfaites statistiquement mais prévisibles. Par ailleurs l'imprévisibilité de suites issues d'un TRNG donné ne peut être établie en analysant les suites générées *a posteriori* : celles-ci ne donnent aucune information sur la manière dont elles ont été générées, *i.e.* de manière déterministe ou aléatoire. L'imprévisibilité effective de la sortie d'un TRNG s'établit uniquement via la modélisation et analyse de son fonctionnement en lien avec des mesures sur le phénomène aléatoire : 1) Le phénomène aléatoire exploité doit être caractérisé, mesuré et modélisé par une loi aléatoire adéquate. 2) Le comportement de l'extracteur doit être modélisé afin de mettre en lien la loi caractérisant le phénomène aléatoire et les nombres en sortie du générateur (qui sont représentés par des variables aléatoires). 3) Enfin, l'étude des variables aléatoires représentant ces nombres binaires permet de calculer des quantités comme l'entropie de Shannon par bit généré. Celle-ci, dans le cas particulier de cette analyse, est interprétée comme une mesure théorique de l'imprévisibilité des bits de sortie. Sa valeur varie entre 0 (nombres déterministes) et 1 (nombres imprévisibles, indépendants et uniformément répartis).

Très peu de travaux de TRNG entreprennent la démarche de caractériser le phénomène aléatoire exploité et de proposer une quantification de l'entropie réelle par bit généré. Dans beaucoup

4. Le sigle TRNG est abondamment utilisé dans la littérature scientifique pour désigner ces générateurs

de cas, les concepteurs valident le fonctionnement de leur générateur uniquement à l'aide de tests statistiques. Certains travaux proposent une modélisation de l'entropie mais rarement en lien avec une mesure et caractérisation rigoureuse de la source d'aléa utilisée. Or, une analyse de sécurité complète d'un TRNG selon les critères contemporains se fait sur plusieurs aspects. La source d'aléa doit être identifiée et rigoureusement caractérisée. L'extraction d'entropie doit être maîtrisée : la modélisation du principe d'extraction, utilisée conjointement avec les mesures sur la source d'aléa, permet de quantifier l'entropie en sortie du générateur. L'établissement d'un modèle de menace lié à l'architecture et principe du générateur est nécessaire afin de proposer les contre-mesures adaptées. Des mécanismes de tests et alarmes internes et des moyens de *monitoring* doivent également être mis en place afin de renforcer la sécurité du générateur face aux attaques et d'éventuelles défaillances de la source d'aléa. Enfin, les séquences générées sont évaluées à l'aide des batteries de tests statistiques. De manière générale et au mieux, le principe doit permettre de quantifier l'entropie en sortie du générateur et de la contrôler à l'aide des paramètres de l'extracteur (mécanisme d'extraction d'aléa). Au pire, le principe doit au moins permettre de quantifier l'entropie en fonction de paramètres mesurés sur la source d'aléa sans forcément la contrôler. L'usage de TRNG qui ne permet ni de contrôler ni de quantifier l'entropie est fortement déconseillé dans les applications cryptographiques.

La principale source d'aléa dans les circuits électroniques est le bruit thermique lié aux mouvements aléatoires des porteurs de charges dans les semi-conducteurs. Celui-ci se manifeste au niveau du signal numérique sous la forme de variations aléatoires (très petites) des temps de propagation des portes logiques, ce phénomène est appelé *jitter* (guigue). L'une des techniques les plus populaires pour générer de l'aléa dans les circuits numériques consiste à capter ces variations dans des signaux oscillants. Plusieurs méthodes existent et différentes structures oscillantes sont utilisées avec chacune leurs avantages et inconvénients (anneaux à inverseurs, boucles à verrouillage de phase, oscillateurs bi-stables, anneaux de Fibonacci/Galois, etc). Cependant, les oscillateurs en anneau à inverseurs ressortent du lot pour leur simplicité, leur bonne intégration dans les flots de conception, et leur faible coût.

Objectifs de la thèse

Plusieurs études ont cherché à étudier et caractériser les oscillateurs dans les circuits numériques en fonction de leur adaptabilité à la génération d'aléa (la qualité du bruit qui les caractérise, la présence ou non d'un principe d'extraction adapté à leur fonctionnement, leur robustesse aux variations de température, tension et à la variabilité de procédé de fabrication, leur précision en fréquence, en phase, etc). Une démarche de caractérisation et de comparaison entre les différents oscillateurs numériques est entreprise dans [Val10].

Ces dernières années ont vu l'émergence d'un nouveau type d'oscillateurs, appelés anneaux auto-séquencés (*Self-timed Rings - STR*). Ces structures, issues du rebouclage de circuits de contrôle de micropipelines asynchrones, ont commencé à être étudiées en tant qu'oscillateurs au début des années 2000. Les travaux présentés entre 2003 et 2010, principalement dans la conférence ASYNC (*IEEE International Symposium on Asynchronous Circuits and Systems*), ont mis en

évidence des caractéristiques prometteuses en terme de re-configurabilité, précision en fréquence, adaptabilité à la génération de signaux muti-phasés et robustesse aux variations de procédé de fabrication. Cependant, il n'existait jusqu'à présent aucun travail visant à étudier l'adaptabilité de ces oscillateurs aux principes de TRNG existants, ou à proposer un nouveau principe adapté au fonctionnement bien particulier de ces anneaux.

Ce travail de thèse est une première exploration d'architectures de TRNG basées sur des anneaux auto-séquencés. Il est né de la collaboration de deux équipes de recherche françaises : CIS (*Concurrent Integrated Systems*) du laboratoire TIMA de Grenoble, avec ses contributions dans la conception de circuits asynchrones, et SES (*Secure Embedded Systems*) du laboratoire LaHC de Saint-Etienne, qui contribue dans le domaine de la cryptographie appliquée, notamment la génération de nombres aléatoires. Les objectifs de la thèse sont les suivants :

1) Proposer un ou des nouveaux principes de génération d'aléa embarqués dans des circuits logiques et utilisant les techniques de conception asynchrone.

2) Ces principes devront être suffisamment rapides, robustes vis-à-vis des attaques et des changements d'environnement (si possible avec une preuve de la robustesse), économiques en terme de surface et de consommation et pratiques en utilisation (sans nécessité d'une intervention manuelle dans la phase de la conception, notamment pour le placement/routage).

3) Proposer un modèle mathématique caractérisant le comportement du générateur et notamment la qualité de la suite binaire aléatoire générée en relation avec les paramètres statistiques de la source d'aléa employée.

4) Proposer des outils et méthodes pour tester en temps réel la qualité de la sortie du générateur.

Contributions et plan du mémoire

La contribution principale de ce travail de thèse est la proposition d'un nouveau principe de TRNG ainsi qu'une méthode pour le dimensionner en fonction de paramètres mesurés sur la source d'aléa, et le sécuriser face aux attaques et défaillances. Ce principe est basé sur l'extraction du *jitter* d'événements (transitions électriques) se propageant au sein d'un anneau auto-séquencé. Nous montrons à travers ce travail que la configuration et le mode de fonctionnement uniques de l'anneau auto-séquencé permettent d'utiliser un nouveau principe de génération d'aléa qui tire avantage des spécificités bien particulières de ces oscillateurs. En particulier, le taux d'entropie en sortie de ce générateur peut être finement réglé en relation avec la configuration de l'anneau auto-séquencé.

Ce document est divisé en 2 parties. La première partie est un état de l'art général sur la conception et évaluation de TRNG (chapitre 1) ainsi que les oscillateurs en anneau auto-séquencés (chapitre 2). La seconde partie traite de la caractérisation, de la modélisation, de l'évaluation et de la sécurisation d'un nouveau principe de TRNG à base d'anneau auto-séquencé. Le chapitre 3 présente le TRNG à base d'anneau auto-séquencé : son principe de fonctionnement, l'architecture de son coeur ainsi que la méthode pour le calibrer en fonction des caractéristiques de bruit dans le circuit. Le chapitre 4 caractérise le *jitter* dans les anneaux auto-séquencés. Ensuite, le chapitre 5 propose une modélisation stochastique du coeur du générateur : un taux d'entropie minimal par bit

de sortie est calculé en fonction des paramètres du bruit (mesurables) et ceux de l'anneau (qu'on peut contrôler). Ce modèle permet le dimensionnent fin du générateur en fonction du niveau de sécurité visé. Ce chapitre propose également une analyse du modèle de menace du générateur et une série d'alarmes et tests embarqués adaptés au principe. Enfin, le dernier chapitre est consacré à l'évaluation du coeur du générateur dans deux familles de FPGA (Altera Cyclone III et Xilinx Virtex 5) et un circuit CMOS 350 nm.

Première partie

Etat de l'art

Chapitre 1.

Générateurs de nombres aléatoires

Sommaire

1. Introduction

Un des principes fondamentaux de la cryptographie moderne stipule que les algorithmes de chiffrement sont publics, leur sécurité étant basée sur le secret de la clé de chiffrement (principe de Kerckhoffs). Comme ces clés sont générées de manière automatisée, elles doivent être imprévisibles et ne doivent pas avoir de défauts statistiques permettant d'améliorer les chances de les deviner autrement qu'en testant toutes les possibilités (attaque par force brute). Les systèmes cryptographiques modernes s'appuient donc largement sur les générateurs de nombres aléatoires pour produire des clés de chiffrement, des vecteurs d'initialisation, des masques jetables, des bits de *padding* et des challenges dans les protocoles d'authentification. Certains systèmes de chiffrement sont même entièrement basés sur l'utilisation de nombres aléatoires comme le chiffrement par masque jetable (*one time pad*).

Un générateur de nombre aléatoire idéal est une construction mathématique qui génère des nombres aléatoires indépendants et uniformément répartis [KS01]. En pratique, les RNG se répartissent en deux classes avec possibilité d'hybridation entre les deux : les générateurs de nombres véritablement aléatoires (*True Random Number Generators - TRNG*) et les générateurs de nombres pseudo-aléatoires (*Deterministic Random Numbers Generators - DRNG*). Les DRNG s'appuient sur des algorithmes déterministes pour générer des nombres qui, en apparence, semblent aléatoires. Leur nature algorithmique fait qu'ils sont facilement implantables dans les circuits numériques et peuvent fonctionner à des débits importants. Cependant, en théorie, les DRNG ne garantissent pas l'imprévisibilité des suites générées. Pour remédier à cela, ils utilisent souvent une graine d'initialisation issue d'un TRNG. Les TRNG ne sont pas de nature algorithmique : il s'agit de constructions électriques ou mécaniques qui extraient l'aléa à partir d'un phénomène le plus souvent physique possédant des propriétés aléatoires (phénomènes quantiques, mouvements chaotiques dans une lampe à lave, bruit atmosphérique pendant un orage ...). Leur débit est limité par le spectre de fréquence du phénomène dont ils tirent l'aléa (fréquence à laquelle une main peut jeter un dé, spectre fréquentiel du bruit électronique ...) et par les contraintes liées à la technique d'extraction et à son implantation. La qualité statistique des suites générées dépend à la fois de la qualité de la source d'aléa et de la technique avec laquelle on l'extrait, cependant des défaut statistiques liés à l'implantation sont souvent observés en pratique. Néanmoins, s'ils sont correctement conçus, les TRNG permettent de générer des suites imprévisibles.

Ce chapitre présente un état de l'art sur les générateurs de nombres aléatoires, en insistant particulièrement sur les TRNG dans les circuits numériques et la manière de les concevoir et de les évaluer. La section 2 décrit brièvement les DRNG : leur fonctionnement, applications, critères de sécurité et les outils pour tester les propriétés statistiques des suites binaires. La section 3 définit les TRNG, leur principe de fonctionnement et présente quelques techniques d'extraction d'aléa que l'on retrouve souvent dans les circuits numériques. La section 4 s'intéresse en détail à la conception et l'évaluation de TRNG selon des critères principalement axés sur la sécurité. Nous proposons par ailleurs d'y analyser quelques principes de TRNG en fonction de ces critères. La section 5 conclut ce chapitre. Enfin, l'annexe A est associée à ce chapitre, elle décrit plusieurs principes de TRNG proposés dans la littérature sans s'intéresser particulièrement à leur sécurité.

2. Générateurs de nombres pseudo-aléatoires (DRNG)

Les générateurs de nombres pseudo-aléatoires (DRNG) génèrent des séquences binaires qui ont les mêmes propriétés statistiques qu'une suite de séquences aléatoires, mais qui en théorie sont prévisibles. Leur sortie est évaluée à l'aide de tests statistiques sur les suites binaires, et leur niveau de sécurité grâce à la cryptanalyse de l'algorithme utilisé. Cette classe de générateurs est facile à implanter et permet des débits importants tout en produisant des suites qui ont de bonnes propriétés statistiques. Elle est donc très adaptée aux applications ne nécessitant pas l'imprévisibilité des suites (comme la simulation numérique), mais peut également être utilisée dans les applications cryptographiques à condition de respecter certains critères.

2.1. Principe et applications

Les séquences générées par un DRNG ne sont pas véritablement aléatoires dans le sens où elles sont complètement déterminées par un ensemble (généralement petit) de paramètres initiaux. L'algorithme utilisé peut donc être complètement caractérisé par la succession déterministe de ses états internes. La période d'un DRNG est définie par le nombre de tous ces états, généralement exprimée en bits. Généralement, il est assez facile de construire des algorithmes avec des périodes suffisamment longues pour la plupart des applications. D'autres part, ces algorithmes sont construits de manière à fournir des nombres qui ont des bonnes répartitions statistiques. Leur facilité d'implantation (possible en logiciel), rapidité et la multitude d'algorithmes efficaces connus font que ce type de générateurs est très utilisé dans les applications comme la simulation numérique (méthodes de *Monte Carlo*), les automates de jeux et mêmes certaines applications cryptographiques ne nécessitant pas l'imprévisibilité des suites. Parmi les DRNG les plus communs, on peut citer le générateur *middle-square* de J. von Neuman [VN51] et les générateurs congruentiels linéaires, introduits par D. H. Lehmer en 1948 [Leh51]. Ces derniers sont devenus particulièrement populaires et répandus depuis leur amélioration par G. J. Mitchell et D. P. Moore en 1958.

Les DRNG peuvent également être utilisées dans des applications cryptographiques nécessitant des suites imprévisibles si leur graine d'initialisation est renouvelée périodiquement à l'aide d'un TRNG (générateur de nombres véritablement aléatoires). Ceci permet de profiter à la fois de la rapidité des DRNG et de l'imprévisibilité des TRNG. Il est important de noter que si un DRNG est toujours initialisé au même état, il produira toujours la même séquence de sortie. Il est donc essentiel que la graine d'initialisation (issue d'un TRNG) soit renouvelée avec une période inférieure à celle de l'algorithme.

2.2. DRNG dans les applications cryptographiques

Dans la littérature, un DRNG adapté aux applications cryptographiques est appelé générateur de nombres pseudo-aléatoires cryptographiquement sûr (*Cryptographically Secure DRNG*). Le critère principal de ce type de DRNG est le suivant : un adversaire qui n'a pas de connaissance de la graine d'initialisation ne doit pas pouvoir distinguer les suites issues du DRNG de suites

véritablement aléatoires. Malgré le fait que cette propriété ne peut être rigoureusement prouvée, une forte évidence peut être apportée par la cryptanalyse, en montrant par exemple que prédire la sortie du DRNG revient à résoudre un problème considéré comme difficile ou insolvable, comme la décomposition en produit de facteurs premiers [SS05]. De plus, ces DRNG doivent être capables de résister à des attaques visant à introduire des imperfections dans l'algorithme (comme l'injection de données). Cette classe de DRNG inclut les algorithmes de chiffrement par flot (*stream ciphers*), certains algorithmes de chiffrement par bloc (*block ciphers with output feedback*) ainsi qu'un certains nombre de DRNG basés sur une hypothèse de complexité de calcul (*Computational hardness assumption*), comme l'algorithme de Micali-Shnorr et le générateur *Blum Blum Shub* [BBS86]. Parmi ces algorithmes, on peut également citer *Yarrow* [KSF99] et *Fortuna* [FS03].

La méthodologie AIS31 [1], proposée par le BSI allemand pour l'évaluation de RNG destinés aux applications cryptographiques, décrit quatre classes pour les DRNG en fonction de leur niveau de sécurité, présentées dans Tab. 1.1. Les critères de classification consistent en deux propriétés qui peuvent éventuellement être établies en analysant l'algorithme et la structure du DRNG : le secret en avant [2] (*forward secrecy*) et le secret en arrière (*backward secrecy*). Le secret en avant signifie qu'il est impossible de déterminer (calculer, ou deviner avec une probabilité non-négligeable) les futures valeurs de la sortie d'un DRNG à partir de sa valeur de sortie courante ou de ses précédentes valeurs produites. On parle de secret en avant amélioré (*enhanced forward secrecy*) lorsque même l'état interne du générateur (pas uniquement ses sorties) ne permet pas de deviner l'évolution future de l'algorithme. De manière similaire, le secret en arrière (*backward secrecy*) signifie qu'il est impossible de déterminer les valeurs précédentes de la sortie du DRNG à partir de la valeur de sortie courante ou des futures valeurs de sortie. Si de plus, l'état interne du DRNG ne permet pas de deviner les valeurs précédentes, alors on parle de secret en arrière amélioré (*enhanced backward secrecy*). Il est à noter que le secret en avant amélioré (*enhanced forward secrecy*) qui caractérise la classe DRG. 4 peut être obtenu en réinitialisant régulièrement les états internes du DRNG à l'aide de graines issues de TRNG (DRNG hybride).

Classe de RNG	Critères
DRG.1	DRNG avec secret en avant (*forward secrecy*) selon ISO 18031
DRG.2	DRG.1 avec secret en arrière (*backward secrecy*) selon ISO 18031
DRG.3	DRG.2 avec secret en arrière amélioré (*enhanced backward secrecy*)
DRG.4	DRG.3 avec secret en avant amélioré (*enhanced forward secrecy*) - DRNG hybride

Table 1.1 – Classes de DRNG définies dans [KS01]

Finalement, en fonction de l'application, les DRNG peuvent dans certains cas être uniquement évalués à l'aide de tests statistiques (DRNG pour la simulation numérique, l'instrumentation). Pour les applications cryptographiques nécessitant l'imprévisibilité des nombres, l'évaluation se fait à l'aide des tests statistiques, mais aussi grâce à la cryptanalyse de l'algorithme (périodicité, preuve

1. "A proposal for : Functionality classes for random number generators", BSI (*Bundesamt für Sicherheit in der Informationstechnik*) [KS01]

2. Le terme *forward secrecy* a un équivalent français qui est "confidentialité persistante", par contre il n'existe pas encore d'équivalent pour le terme *backward secrecy*

de la complexité de prédire ses suites ...). En pratique, leur sécurité dépend donc : 1) du secret de la graine d'initialisation (celle-ci peut être générée à l'aide d'un TRNG), 2) de la périodicité de l'algorithme (plus elle est longue, plus le DRNG est sûr, et plus la période de renouvellement de sa graine peut être grande) 3) et enfin des preuves de sécurité apportées par la cryptanalyse de leur fonctionnement (liées à l'algorithme en lui-même).

2.3. Evaluation statistique des suites binaires

La qualité statistique est un critère primordial pour valider les suites issues d'un RNG, celles-ci ne doivent pas comporter de faiblesses statistiques (biais, périodicité ...). Un test statistique est un outil décisionnel qui permet de vérifier si une suite binaire possède les mêmes propriétés statistiques qu'une suite aléatoire (uniformité, indépendance, etc.). Le principe de ces tests consiste à comparer des propriétés statistiques des suites testées avec celle d'une suite (théorique) qui aurait été générée par un RNG idéal.

Bien qu'ils soient indispensables pour valider le fonctionnement des RNG, il est très important de préciser que les tests statistiques ne permettent absolument pas de vérifier l'imprévisibilité des suites générées. Celle-ci ne peut être rigoureusement établie qu'en ayant connaissance de la manière dont ont été générées ces suites et par une analyse rigoureuse du fonctionnement interne du générateur qui les a produit (ceci est discuté dans la section traitant des TRNG). Par contre, ces tests permettent de déceler et de quantifier les défauts statistiques tels que la non-uniformité des suites générées et la présence de dépendances. Ils restent par ailleurs essentiels dans l'évaluation des algorithmes utilisés par les DRNG.

Soit B un ensemble fini constitué de 2 symboles : $B = \{0, 1\}$. D'après la définition donnée dans [Mau92], un test statistique T est une fonction mathématique prenant en entrée une séquence s constituée de N symboles appartenant à B, et fournissant en sortie un booléen (*accept* ou $reject$) :

$$T : B^N \rightarrow \{accept, reject\}$$

Le principe du test statistique consiste généralement à rejeter ou à accepter une hypothèse statistique, appelée hypothèse nulle (H_0). En fonction de celle-ci, il existe plusieurs types de tests statistiques. Les tests de conformité consistent à comparer un paramètre donné de la séquence s (par exemple le biais) avec une valeur pré-déterminée. Les tests d'adéquation vérifient la compatibilité de la séquence s avec une distribution donnée. Les tests d'homogénéité vérifient si plusieurs sous-séquences de s proviennent de la même population. Enfin, les tests d'indépendance cherchent à déceler des associations entre sous-séquences de s.

Il existe plusieurs batteries de tests statistiques dans la littérature scientifique. Les plus utilisés dans le domaine de la cryptographie sont NIST SP 800-22 [RSN+01] et FIPS 140 [Lab94], tous deux proposés par l'organisme de standards technologiques américain NIST (*National Institute of Standards and Technology*), ainsi que DIEHARD, proposé par des universitaires.

2.3.1. Les tests FIPS 140

A l'origine, FIPS 140 (*Federal Information Processing Standards*) est une norme gouvernementale américaine, créée en 1994 par le NIST, qui spécifie les exigences de sécurité pour les équipements cryptographiques. La section traitant de la gestion et génération de clés pour les applications cryptographiques propose une suite de 4 tests statistiques pour évaluer la qualité de ces clés (FIPS 140-1 [Lab94]). La mise à jour de la norme en 2001 (FIPS 140-2 [Lab01]) modifie plus sévèrement les critères et seuil de passage de ces tests. Cependant, ils ont été tout simplement retirés dans une révision datant de 2006.

Les tests proposés dans FIPS 140 prennent en entrée une séquence de 20000 bits et donnent un résultat binaire en sortie : réussite ou échec. Ces tests sont au nombre de quatre : *Monobit* (test de biais), *Poker* (test d'indépendance), *Runs* et *Long runs* (détectent de longues suites de '0' et de '1').

Bien qu'elle soit loin d'être complète et qu'elle ne traite qu'un nombre limité d'échantillons, la suite de tests statistiques FIPS 140 est un bon point de départ pour évaluer rapidement une suite binaire. Généralement, elle est utilisée en conjonction avec une suite plus complète (comme NIST SP 800-22 ou DIEHARD) car elle permet d'éliminer d'emblée et rapidement des séquences n'ayant pas l'air aléatoires avant une analyse plus approfondie et donc plus longue.

2.3.2. Les tests NIST SP 800-22

Dès 1997, le RNG-TWG (*Random Number Generation Technical Working Group*), section de l'organisme gouvernemental NIST, initie un travail de fond ayant 3 objectifs principaux : (a) développer une batterie de tests statistiques pour détecter des séquences non-aléatoires issues de générateurs de nombres aléatoires utilisés dans les applications cryptographiques (b) produire la documentation et une implantation logicielle de ces tests, et (c) fournir une méthodologie pour l'utilisation et application de ces tests.

La batterie de tests statistiques NIST SP 800-22 comporte 16 tests statistiques à l'origine, ce nombre a été réduit à 15 dans une révision datant de 2001 ([RSN+01]). La dernière révision NIST SP 800 rev-1a a été publiée en Avril 2010. Bien que cette suite de tests soit à ce jour la plus complète et la plus évoluée, les auteurs préviennent qu'aucun test statistique ne permet de certifier qu'un générateur est approprié pour une application cryptographique donnée : ces tests ne doivent pas se substituer à la cryptanalyse, ils permettent néanmoins d'éliminer les candidats qui ne sont pas adaptés à une application donnée.

Le principe de ces tests consiste à examiner deux hypothèses : l'hypothèse nulle (H_0), qui est que la suite examinée est aléatoire, et l'hypothèse alternative (H_1), qui est que la suite examinée n'est pas aléatoire. Les seuils de confiance sont définis en fonction des deux types d'erreurs rencontrées dans une telle évaluation : l'erreur de premier ordre α, la probabilité qu'une suite réellement aléatoire ne soit pas reconnue comme tel par le test statistique, et l'erreur de second ordre β, la probabilité qu'une suite non-aléatoire soit reconnue comme aléatoire par le test statistique.

Chaque test est basé sur le calcul d'une quantité statistique qui est fonction des données et du type de test. Si on note cette quantité S et sa valeur critique t, alors l'erreur de premier ordre α

est $P(S \leq t \ \| \ H_0 \ est \ vraie) = P(rejeter \ H_0 \mid H_0 \ est \ vraie)$. Les résultats du test sont donnés sous forme de p_{values}, qui résume la force de l'évidence contre l'hypothèse nulle. Pour chaque test, chaque p_{value} est la probabilité qu'un RNG parfait ait produit une séquence moins aléatoire que la séquence testée au vu du paramètre statistique évalué par le test. Si $p_{value} \geq \alpha$, alors l'hypothèse nulle est acceptée (la séquence paraît aléatoire). α désigne alors le seuil de confiance du test, sa valeur est typiquement comprise entre 0.001 et 0.01 dans le cadre d'applications cryptographiques. Une valeur de α égale à 0.001 indique qu'une séquence aléatoire parmi 1000 testées ne devrait pas avoir l'air aléatoire alors qu'elle l'est réellement. Si $p_{value} \geq 0.001$, la séquence est considérée aléatoire à 99.9% de chances. Si $p_{value} < 0.001$, la séquence est considérée non aléatoire à 99.9% de chances également.

Chaque test requiert plus ou moins de données pour son application, les plus exigeants nécessitant 10^9 bits (comme le *Maurer universal test*), une analyse complète requiert autour de 12 Mo de données pour un seuil de confiance de 0.01.

3. Générateurs de nombres véritablement aléatoires (TRNG)

Les générateurs de nombres aléatoires vrais (TRNG) sont des dispositifs mécaniques ou électriques permettant d'extraire des nombres aléatoires à partir d'une source d'aléa physique. S'ils sont correctement conçus et implantés, même une connaissance absolue de leur architecture et fonctionnement interne ne doit pas permettre de prédire leurs bits de sortie. Dans cette section, nous nous intéressons à leur principe de fonctionnement et à quelques techniques d'extraction d'aléa que l'on retrouve souvent dans les circuits numériques.

3.1. Principe et fonctionnement

Il nous semble important, dans le contexte de cette thèse, de préciser rigoureusement le concept d'aléa, associé aux TRNG, par opposition au hasard. Le hasard peut-être décrit comme la rencontre de deux séries causales indépendantes [3]. Selon cette définition, le hasard peut être associé à la théorie du chaos : celle-ci traite de systèmes dynamiques rigoureusement déterministes, mais qui sont sujet à un phénomène fondamental d'instabilité (appelé "sensibilité aux conditions initiales"), qui, associé à une contrainte de récurrence, rend ces systèmes extrêmement difficiles à prévoir. Les phénomènes hasardeux et chaotiques sont nombreux dans l'univers, mais le vrai aléa est fondamentalement mystérieux, élusif, et difficile à observer : un phénomène est dit aléatoire s'il n'existe aucun observateur qui puisse prédire son évolution future quelque soient les données dont il dispose (par exemple le fait qu'un photon traverse ou non une lame semi-transparente, les fluctuations quantiques du vide [4]).

3. Définition donnée par le mathématicien français Antoine-Augustin Cournot au 19^{ieme} siècle

4. Un TRNG exploitant les fluctuations quantiques du vide est proposé dans : http ://photonics.anu.edu.au/qoptics/Research/qrng.php

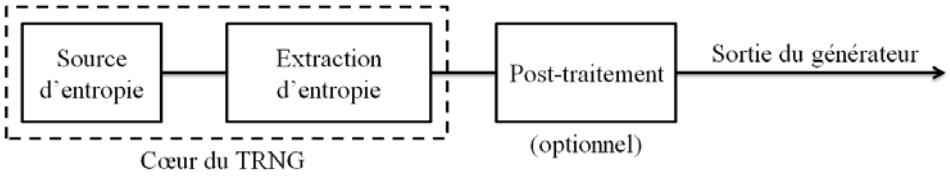


Figure 1.1 – Schéma de principe d'un TRNG

Les TRNG génèrent donc des nombres aléatoires à partir d'un phénomène (le plus souvent physique) dont on ne peut prédire l'évolution quelque soient les connaissances dont on dispose. Un schéma générique de TRNG est montré à la Fig. 1.1. Il est à noter qu'en fonction des principes de TRNG, ces trois parties ne sont pas nécessairement distinctes d'un point de vue architectural. La source d'entropie correspond au processus qui produit l'aléa (le plus souvent physique). L'extraction d'entropie est le mécanisme qui permet d'extraire les propriétés aléatoires de la source pour générer des nombres aléatoires (sous forme de séquences de bits dans le cas des TRNG implantés dans les circuits numériques). Comme les sources physiques d'aléa et les mécanismes d'extraction ne sont jamais parfaits, les nombres en sortie d'un TRNG souffrent souvent de défauts statistiques. Des algorithmes de post-traitement sont alors utilisés pour corriger ces défauts.

3.2. Techniques d'extraction d'aléa dans les circuits numériques

Les vraies sources d'aléa sont extrêmement limitées à l'intérieur des circuits, et semblent toutes *a priori* liées au bruit. Celui-ci est omniprésent dans les circuits électroniques, il a des origines diverses et se présente donc sous différentes formes : bruits de diffusion (bruit thermique, bruit quantique, bruit d'électrons chauds), bruits en excès (bruit de scintillation, bruit de génération/recombinaison, bruit en créneau) et bruits de jonction des semi-conducteurs (bruit de grenaille, bruit d'avalanche). Dans un circuit électronique, le bruit influe sur les signaux manipulés de différentes manières (fluctuations de la tension du signal analogique, *jitter* ...). Cette section présente quelques exemples de techniques permettant d'extraire ces propriétés afin de générer des séquences de bits aléatoires.

3.2.1. Amplification du bruit d'un composant analogique

Certains principes de TRNG utilisent le bruit d'un composant analogique comme une résistance ou une diode Zener comme source d'aléa. Ces techniques cherchent à amplifier les fluctuations de la tension de sortie d'un signal analogique (issu de la source de bruit) et à les capter grâce à des composants analogiques (amplificateurs opérationnels, comparateurs ...). Par exemple, la technique utilisée dans un TRNG développé par *Intel*, dont une évaluation par la société *Cryptography Research* est proposée dans [JK99], consiste à amplifier la différence de tension entre deux sources de bruit (résistances) pour alimenter un VCO (oscillateur contrôlé en tension), celui-ci est

échantillonné à intervalles réguliers pour générer des nombres aléatoires. La structure différentielle utilisant deux sources de bruit indépendantes permet de rendre l'architecture plus robuste aux perturbations externes et fluctuations environnementales. Le schéma de principe de cette technique d'extraction d'aléa est illustré dans Fig. 1.2

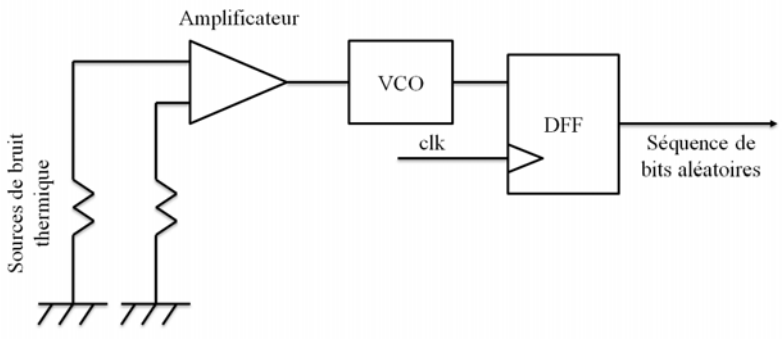


Figure 1.2 – Schéma de principe du TRNG discuté dans [JK99]

Parmi les principes de TRNG publiés basés sur l'amplification du bruit de composants analogiques on peut également citer [BB99] et [KS08]. Le principal défaut de ces techniques est qu'elles font appel à des composants analogiques : elles nécessitent un flot de conception mixte analogique/numérique et ne peuvent pas être implantées facilement dans les circuits re-configurables. Par ailleurs, les débits sont assez faibles (au mieux de l'ordre de la centaine de Ko/s dans [EHK+03], [BB99] et [KS08]).

3.2.2. Extraction du *jitter*

Le *jitter* correspond à de courtes fluctuations temporelles du signal numérique dues au bruit dans les composants électroniques. Il peut être mesuré sur un signal numérique dans le domaine fréquentiel ou temporel (statistiques sur des mesures temporelles), et peut être capté par comptage ou échantillonnage. Les signaux manipulés peuvent être générés à l'intérieur du circuit, et l'extraction se base uniquement sur des composants numériques (bascules D, *latchs*, compteurs ...). Ces techniques sont par conséquent très utilisées dans les circuits numériques.

Traditionnellement, il existe deux méthodes pour générer des nombres aléatoires à partir du *jitter* d'un signal oscillant : échantillonner un signal numérique au plus près de son moment de basculement (la valeur échantillonnée dépend alors du *jitter* comme cela est montré dans Fig. 1.3), ou bien compter le nombre d'oscillations d'un signal oscillant sur une fenêtre de temps donnée (les variations temporelles dues au *jitter* sont additives dans le temps et peuvent influer sur le nombre d'oscillations si la fenêtre de temps est suffisamment longue, on parle alors d'accumulation du *jitter*).

La technique proposée dans [FMC85] exploite la propriété d'accumulation du *jitter* dans le temps. L'architecture utilise une simple bascule D (*flip-flop*) qui échantillonne un signal *jitté* pour en extraire les nombres aléatoires. Cette technique sera dorénavant un des principes de base lar-

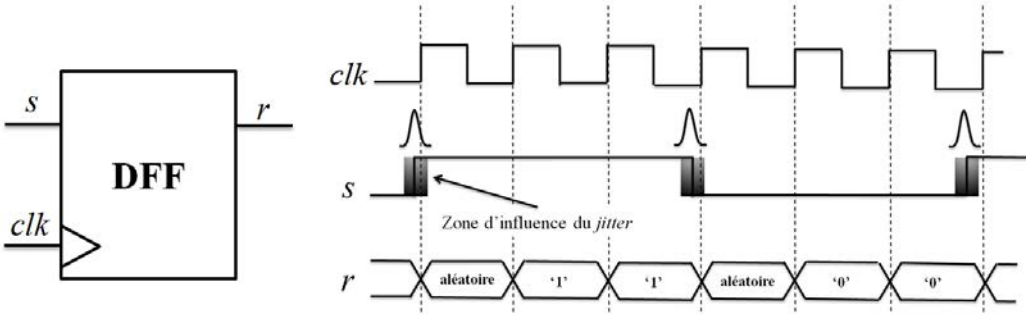


Figure 1.3 – Echantillonnage d'un signal numérique sujet au *jitter* par un signal d'horloge idéal

gement repris et améliorés par la communauté scientifique ultérieurement. Le principe de fonctionnement est décrit dans Fig. 1.4. Dans [FMC85], un signal haute fréquence, généré grâce à un oscillateur embarqué de 8 MHz, est échantillonné par un signal basse fréquence ajustable grâce à des composants externes. La valeur échantillonnée dépend des variations temporelles accumulées sur le signal haute fréquence, ainsi que de celles du signal basse fréquence. Les principaux défauts de cette technique sont la faiblesse du débit et le fait que les variations dues aux bruits déterministes s'accumulent plus rapidement que celles dues aux bruits aléatoires [FBBV08], ce qui peut induire des défauts statistiques importants en sortie. D'autre part le bruit dans les basses fréquences est souvent corrélé à la valeur de la fréquence (on parle de *flicker noise*, aussi appelé bruit en $1/F$) ce qui peut induire des dépendances dans les bits générés en sortie.

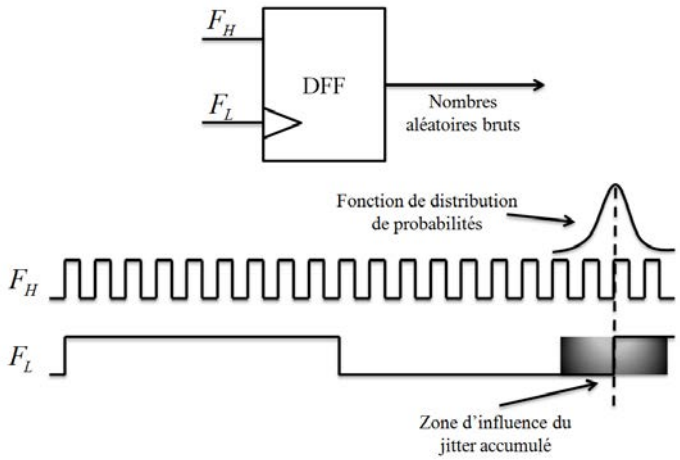


Figure 1.4 – Principe de fonctionnement du TRNG proposé dans [FMC85]

Ce schéma est repris dans [FD02] mais en introduisant une idée nouvelle : en choisissant judicieusement le rapport entre les fréquences des deux signaux oscillants, il est possible d'effectuer un balayage du signal échantillonné avec une résolution temporelle suffisamment fine pour détecter

le *jitter*. Les bits en sortie de la bascule consistent alors en une suite de bits déterministes avec des bits aléatoires intercalés entre eux. En connaissant le rapport entre les fréquences, il est possible d'extraire les bits aléatoires à l'aide d'un décimateur. Ce schéma permet d'éviter d'avoir recours à un signal d'échantillonnage très basse fréquence avec les inconvénients que cela implique (mauvaise qualité du *jitter* et faible débit). Cependant, elle nécessite de maîtriser précisément le rapport entre les fréquences d'oscillateurs. [FD02] présente un *design* permettant de réaliser ce réglage précisément à l'aide d'une PLL (*Phase-locked Loop*).

Une autre technique très populaire et permettant des débits encore plus importants consiste à combiner les sorties de plusieurs signaux indépendants afin de maximiser l'influence du *jitter* sur le signal théorique résultant. L'idée consiste à produire plusieurs transitions électriques (événements) dans des temps très courts en combinant un nombre important de signaux périodiques avec un opérateur XOR. Plus ces signaux sont nombreux, plus il y a d'événements dans une période d'oscillation du signal résultant, et plus ceux-ci sont proches dans le temps de manière à ce que l'influence du *jitter* recouvre une partie de plus en plus importante de la période d'oscillation. Une illustration de ce principe est donnée dans Fig. 1.5 (la zone d'influence du *jitter* est délibérément exagérée dans la figure pour faciliter la compréhension). Cette technique est utilisée dans les TRNG étudiés dans [SMS07] et [WT08], et dans une moindre mesure dans [Gol06]. Il faut cependant préciser que les *designs* actuels utilisant cette technique présentent un certain nombre de failles de sécurité (ceci est discuté dans la section 4.7).

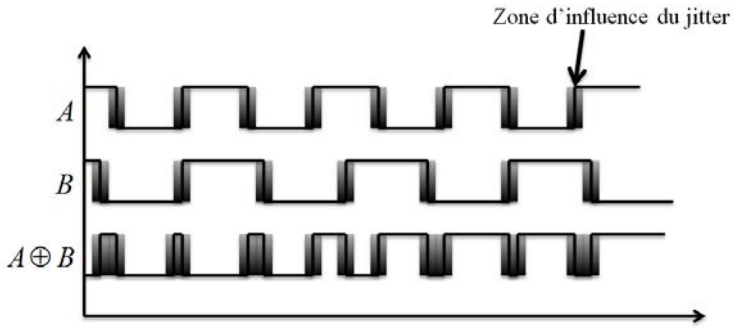


Figure 1.5 – Somme (XOR) de deux signaux *jittés* indépendants

3.2.3. Résolution d'états métastables

En électronique, la métastabilité désigne la capacité d'un système électronique à persister un temps indéterminé dans un état d'équilibre instable [CM73]. Par exemple, en rebouclant la sortie d'une porte inverseuse à son entrée, la tension de sortie peut stagner sur une valeur intermédiaire entre le niveau logique haut et le niveau logique bas ([VHKK08], [EHK+03]). Ce phénomène intervient également (de manière probabiliste) lorsque les temps de *setup* et de *hold* d'une porte logique (notamment des bascules) ne sont pas respectés, la tension de sortie peut alors stagner sur une valeur intermédiaire pendant un temps indéterminé et aléatoire (qui dépend en fait du

bruit) avant d'atteindre sa valeur finale [DGH07]. Enfin, on peut également parler d'oscillations métastables pour désigner un régime d'oscillation transitoire amorti (par exemple dans un circuit bi-stable [VD10]).

Les techniques exploitant la métastabilité pour générer des nombres aléatoires ont pour principe commun de placer un système électronique dans une situation telle que le bruit influence grandement sa résolution. Par exemple, dans [EHK+03], les auteurs provoquent l'état de métastabilité de portes inverseuses (en les rebouclant sur elle même) pour initialiser de manière aléatoire un circuit bi-stable. Ce schéma est décrit dans Fig. 1.6. Les deux multiplexeurs permettent de basculer entre deux modes de fonctionnement. Si le sélecteur est à '0', le circuit consiste en deux inverseurs rebouclés sur eux mêmes. Selon les auteurs, ces inverseurs présentent alors une tension de sortie proche de $V_{dd}/2$ (de notre point de vue, ceci est difficile à obtenir car la structure composée d'un multiplexeur et d'un étage inverseur a plutôt tendance à osciller). Lorsque le sélecteur est à '1', alors les inverseurs sont connectés entre eux et forment un circuit bi-stable dont l'initialisation dépend de la résolution de l'état métastable des inverseurs (influencé en partie par le bruit aléatoire, mais malheureusement aussi par toutes sortes d'influences déterministes). En pratique, l'architecture du TRNG proposé dans [EHK+03] est composée de 15 instances de la cellule de base décrite dans Fig. 1.6. Leurs sorties sont combinées à l'aide d'un XOR pour fournir les nombres aléatoires avant le post-traitement.

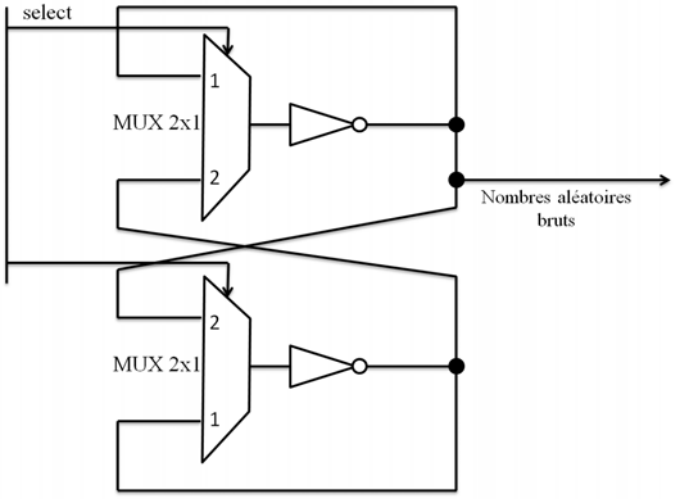


Figure 1.6 – Structure d'une cellule du TRNG proposé dans [EHK+03]

Parmi les techniques exploitant la résolution de situations métastables on peut également citer [VHKK08] et [MKD11]. La difficulté principale avec ces techniques est qu'il est très difficile de quantifier clairement l'influence du bruit sur la résolution des états métastables. D'autre part, il est souvent difficile d'obtenir l'état métastable de manière périodique [Fis12]. Par exemple, dans des implantations concrètes du TRNG présenté dans [EHK+03], la structure composée d'un multiplexeur et d'un inverseur peut parfois osciller (à défaut d'avoir une tension de sortie autour de $V_{dd}/2$ caractéristique de l'état métastable).

4. Conception et évaluation des TRNG

Il est très surprenant de constater que malgré le fait que les TRNG soient des composants très bas-niveau dans un système cryptographique (un TRNG faible peut menacer la sécurité de tout le système), les travaux de TRNG qui s'intéressent rigoureusement à des aspects liés à leur sécurité sont rares. Dans beaucoup de cas (par exemple [Tka03] et [WT08]), leur sortie est évaluée uniquement à l'aide de tests statistiques. Or, la spécificité recherchée des TRNG, qui est l'imprévisibilité de leur sortie, ne peut être établie en analysant les suites générées *a posteriori*. Les méthodes d'évaluation ont beaucoup évolué en ce sens ces dernières années. Dans cette section, nous nous intéressons à la conception et à l'évaluation des TRNG en mettant l'accent sur leur sécurité selon les critères modernes. Nous discutons ensuite quelques principes de TRNG proposés dans la littérature en se basant sur ces critères.

4.1. Conception et évaluation faible de TRNG

Au début des années 2000, les générateurs de nombres aléatoires faisaient partie des rares primitives cryptographiques qui ne disposaient pas de critères d'évaluation standards auprès de l'ITSEC (*Information Technology Security Evaluation Criteria*) ou la CC (*Common Criteria*). Typiquement, dans un certain nombre de travaux anciens et même récents, un principe de TRNG est proposé selon un cahier des charges axé sur les performances (débit, consommation ...), les séquences sont ensuite validées à l'aide de tests statistiques appliqués en sortie d'un post-traitement, ce schéma est illustré dans Fig. 1.7. Parfois le post-traitement consiste en une fonction cryptographique lourde qui masque complètement la qualité de la source (AES, fonctions de hachage...). Souvent, le coeur du TRNG n'est pas clairement spécifié (notamment, la source d'aléa peut être imbriquée dans le mécanisme d'extraction).

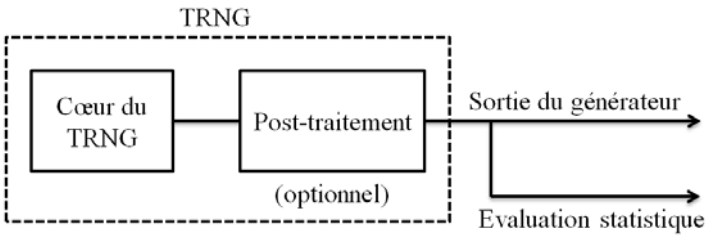


Figure 1.7 – Approche faible d'un point de vue sécurité pour la conception de TRNG dans les circuits numériques

Or, dans le contexte des TRNG, on peut faire l'analogie avec le principe de Kerckhoffs applicable à la sécurité des algorithmes de chiffrement : la sécurité d'un TRNG doit être entièrement basée sur l'imprévisibilité de la source d'entropie (la structure du TRNG étant parfaitement connue et maîtrisée). Malheureusement, une idée reçue a été, pendant longtemps, que les TRNG dans les circuits numériques devaient consister en une association de blocs complexes dont le fonctionnement interne est obscur (mais influencé par le bruit), et qu'il suffisait d'évaluer leur sortie à l'aide de

tests statistiques uniquement en considérant tout le dispositif comme une boite noire. Par exemple, dans [Tka03], Tkacik *et al.* proposent un TRNG utilisant le *jitter* (variations temporelles dues au bruit) de signaux oscillants comme source d'aléa. L'architecture est décrite dans Fig. 1.8. Les deux oscillateurs cadencent deux DRNG consistant en un LFSR (*Linear Feedback Shift Register*) de 43 bits et un automate cellulaire CASR (*Cellular Automata Shift Register*) de 37 bits. Le TRNG fournit des mots de 32 bits à la demande, ceux-ci sont formés en sélectionnant 32 bits de chaque DRNG et en les ajoutant bit à bit en base 2 (XOR). Les suites générées sont ensuite évaluées avec les tests DIEHARD, FIPS 140-1 et Crypt-X pour "valider" le fonctionnement du générateur.

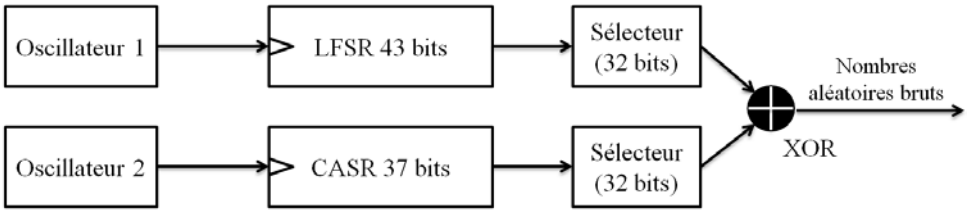


Figure 1.8 – Architecture du TRNG proposé dans [Tka03]

Ce type d'architecture est assez emblématique de la problématique d'évaluation des TRNG au début de la décennie et du fait que l'analyse statistique n'est pas suffisante pour certifier leur sécurité. Les différents choix sur la structure du générateur ne sont pas justifiés et relèvent presque de l'association hasardeuse (LFSR, CASR, leur taille, taille des sélecteurs ...). Le mécanisme d'extraction d'aléa semble imbriqué dans les automates cellulaires, de sorte qu'il n'est pas possible de faire le lien entre l'imprévisibilité effective de la sortie et la source d'aléa (celle-ci est juste mentionnée dans l'étude, aucune mesure n'est donnée). Le comportement pseudo-déterministe masque complètement les caractéristiques de la source. D'ailleurs, dans [Dic03], l'auteur mentionne qu'en déterminant les fréquences des deux oscillateurs, il est possible de prédire la sortie de ce TRNG avec une probabilité importante.

Une telle approche, consistant à combiner une source d'aléa dont les propriétés ne sont pas connues à des mécanismes d'extraction complexes et obscurs, n'est évidemment pas acceptable pour les applications cryptographiques. Le principal critère d'un TRNG doit toujours être en premier lieu l'entropie (l'imprévisibilité de sa sortie). Les résultats d'analyse statistique et le débit sont des facteurs secondaires : les défauts statistiques peuvent être corrigés et le débit relevé (en utilisant le TRNG pour générer des graines d'un DRNG) alors que l'imprévisibilité (entropie) ne peut être créée artificiellement.

4.2. Conception de TRNG avec la sécurité renforcée

Dès 2001, la BSI (*Bundesamt für Sicherheit in der Informationstechnik*) propose une première version du document AIS31 (*"A proposal for : Functionnality classes for random number generators"*, [KS01]) qui établit une méthodologie pour évaluer et classifier les RNG en fonction de critères

basés sur leur nature, sécurité et testabilité, et définit le cadre applicatif d'utilisation de ces différentes classes. Ce document a été mis à jour en 2011 ([KS11]) avec une nouvelle classification plus complète et détaillée.

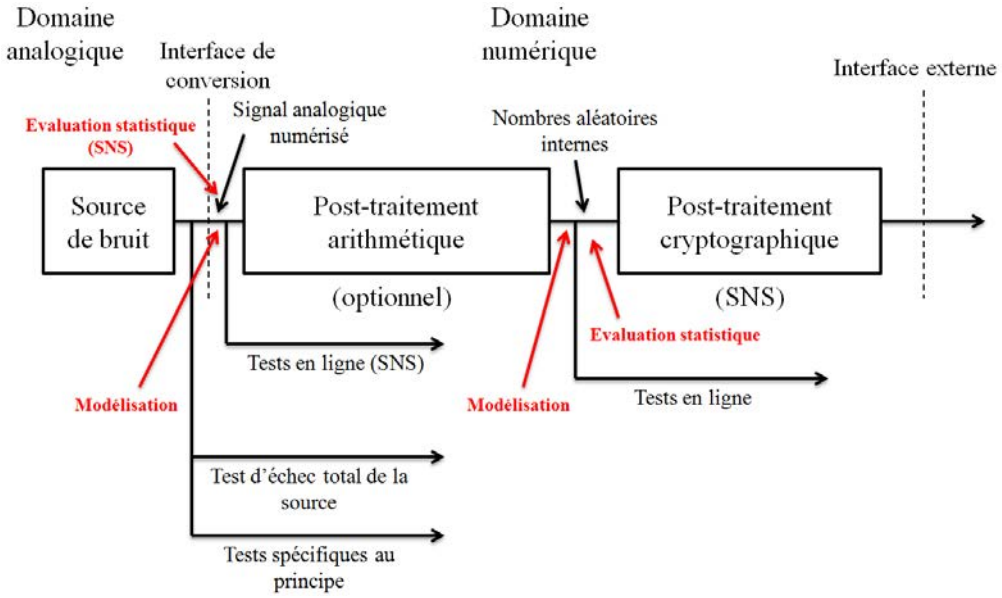


Figure 1.9 – Schéma d'un TRNG avec la sécurité renforcée selon AIS31 (SNS = Selon le Niveau de Sécurité exigé par l'application)

Du point de vue de la conception des TRNG, AIS31 préconise un schéma permettant la testabilité interne du TRNG via divers points d'accès dans son architecture (source d'aléa, signaux internes ...), comme cela est montré dans Fig. 1.9. Une alarme d'échec total permet de détecter une défaillance grave de la source d'entropie. Les tests en ligne permettent de détecter une défaillance statistique non tolérable des nombres aléatoires internes (il ne s'agit pas forcément d'effectuer une évaluation statistique complète des suites générées). Ces tests sont effectués en ligne à la demande et ne sont généralement pas embarqués dans le matériel. En fonction du niveau de sécurité visé, ils peuvent être appliqués avant ou après un post-traitement arithmétique (celui-ci a pour but de corriger d'éventuels défauts statistiques). Les tests spécifiques au principe de génération d'aléa utilisé dépendent de la nature de la source d'aléa et de la mise en oeuvre de l'extraction d'entropie et sont donc *a priori* spécifiques à une famille de TRNG donnée. Ces tests sont embarqués dans le matériel et doivent s'appliquer au plus près de la source d'aléa. Il est à noter qu'une partie importante de AIS31 traite de la modélisation stochastique des générateurs qui est un critère essentiel pour obtenir des niveaux de sécurité suffisants pour la plupart des applications cryptographiques. Celle-ci est traitée dans la section 1.4.3. Enfin, pour obtenir les plus haut niveaux de sécurité, le TRNG doit être associé à un post-traitement cryptographique qui puisse prendre le relais temporairement en cas de défaillance de la partie physique du générateur. En fonction de la présence de modèle, du niveau de testabilité et de la présence ou non de post-traitement cryptographique, la méthodologie définit trois classes de TRNG physiques (*i.e.* utilisant un processus physique comme

source d'entropie), leurs critères de sécurité ainsi que leurs domaines d'application. Ces classes de TRNG sont discutées dans la section 4.6.

A notre connaissance, il n'existe aucun travail publié dans la littérature scientifique qui prenne en compte toutes les recommandations de AIS31, notamment en ce qui concerne l'implantation des tests internes. Le travail qui se rapproche le plus de ce schéma de conception est présenté dans [KS08]. Cependant, on peut noter qu'il existe un certain nombre de TRNG pour lesquels ce schéma de conception et d'évaluation semble parfaitement applicable ([FMC85], [FD02], [VD10], etc).

L'approche utilisée dans l'équipe SES (*Secured Embedded Systems*) du Laboratoire Hubert Curien est une extension du schéma de conception et évaluation d'AIS31, proposée dans [Fis12], dont la spécificité principale est la mesure embarquée de la source d'aléa afin de permettre un *monitoring* en temps réel de l'entropie, ou au moins à la demande. Ceci peut être réalisée si un certain nombre de conditions sont réunies. La source d'aléa doit être mesurable en interne : par exemple, le *jitter* peut être quantifié à l'aide de mesures statistiques sur des grandeurs temporelles (délai de propagation, période d'oscillation ...). Le fonctionnement des différents blocs du *design* doivent être suffisamment maîtrisés pour permettre une modélisation de l'extraction d'entropie en lien avec les paramètres du *design* et ceux de la source. La précision de la mesure embarquée n'est pas forcément déterminante dans l'optique où elle doit surtout permettre la mise en place de seuils d'alarmes. Par contre, elle doit être effectuée à l'endroit même ou a lieu l'extraction d'entropie, tout en influençant le moins possible son fonctionnement. L'idée principale de cette approche est donc de détecter d'éventuelles attaques et défaillances au plus près de la source d'entropie (par exemple une attaque électromagnétique qui chercherait à influer sur la source d'entropie).

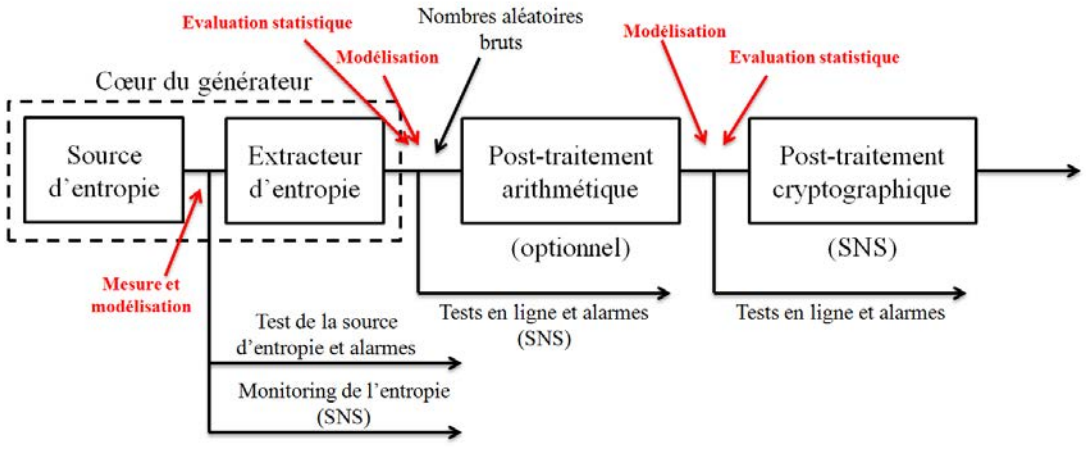


Figure 1.10 – Approche utilisée dans l'équipe SES pour la conception de TRNG dans les circuits numériques

4.3. Modélisation des TRNG et estimation d'entropie

La modélisation des TRNG est nécessaire car c'est le seul moyen d'établir l'imprévisibilité effective de leur sortie. Son objectif est de faire le lien entre l'imprévisibilité de la source d'aléa et celle des bits en sortie du TRNG, elle nécessite donc de bien connaître la source et de parfaitement maîtriser le principe d'extraction.

La notion mathématique qui représente efficacement les expériences aléatoires est celle de variable aléatoire. Une variable aléatoire X est une grandeur qui peut prendre différentes valeurs avec différentes probabilités [5]. Elle peut associer à chaque éventualité une valeur quantitative (un nombre), qualitative (couleurs, symboles), ou même une fonction, dans ce cas on parle de processus stochastique. Chaque éventualité (ou réalisation) de la variable aléatoire est associée à une probabilité d'occurrence donnée. La fonction qui associe, à chaque valeur ω de Ω (l'ensemble des valeurs que peut prendre X) une probabilité $P(X = \omega)$ (la probabilité que X prenne la valeur ω) est appelée loi de probabilités de la variable aléatoire X (ou distribution de probabilités). Prenons par exemple le cas du jet d'une pièce de monnaie idéale (sans défaut). Cette expérience aléatoire peut être décrite par une variable aléatoire X qui prend ses valeurs dans un ensemble $\Omega = \{p, f\}$ avec $P(X = p) = P(X = f) = 1/2$.

La notion d'entropie est souvent associée aux variables aléatoires comme une mesure de leur incertitude. Notons les variables aléatoires avec des lettres majuscules et leurs réalisations (i.e. les valeurs que prennent ces variables) avec des lettres minuscules. Soit X une variable aléatoire prenant ses valeurs dans un ensemble Ω (fini ou infini, mais discret, par exemple $\Omega = \mathbb{N}$). On définit l'entropie de Shannon [6] associée à la variable aléatoire X :

$$H(X) = -\sum_{\omega \in \Omega} P(X = \omega) log_2(P(X = \omega)) \tag{1.1}$$

Il est intéressant de remarquer que l'entropie est à l'origine utilisée en théorie de l'information comme une mesure de l'incertitude contenue dans un message. Une séquence de bits étant représentée par une variable aléatoire X prenant ses valeurs dans $\Omega = \{0, 1\}^n$, l'entropie représente le nombre de bits nécessaires en moyenne pour représenter les réalisations de X. L'entropie est maximale (égale à n) lorsque toutes les valeurs sont équiprobables. Pour une expérience aléatoire quelconque, on pourrait l'interpréter de manière candide comme une mesure de l'aléa. Par exemple, dans le cas du jet d'une pièce de monnaie, si les éventualités sont équiprobables ($P(X = p) = P(X = f) = 1/2$), alors l'entropie est égale à 1. Si les probabilités ne sont pas équilibrées, l'entropie a une valeur comprise entre 0 et 1, d'autant plus faible que les probabilités s'éloignent de 1/2. Si la pièce, tombe toujours du même côté, l'entropie est nulle.

L'objectif principal d'une modélisation stochastique de TRNG est donc de quantifier l'entropie par bit de sortie du générateur, ou du moins en estimer une limite basse. Ces bits de sortie sont modélisés par des variables aléatoires prenant leurs valeurs dans l'ensemble $\Omega = \{0, 1\}$, l'entropie

5. Plus rigoureusement, étant donné un espace probabilisé $\{\Omega, \mathcal{F}, \mathbb{P}\}$ et un espace mesurable $\{E, \mathcal{E}\}$, on appelle variable aléatoire de Ω vers E toute fonction mesurable X de Ω vers E

6. Dans la suite, selon la convention commune, l'entropie de Shannon sera seulement désignée par le terme entropie

par bit de sortie du générateur a donc une valeur comprise entre 0 et 1. Elle est généralement interprétée, dans le cas particulier de cette étude, comme un indicateur de l'aléa : une entropie égale à 0 indique que les bits générés sont déterministes, tandis qu'une entropie non nulle indique que les bits sont plus ou moins imprévisibles (plus celle-ci se rapproche de 1 moins les bits sont biaisés et plus ils sont indépendants).

Le modèle stochastique peut dans un premier temps s'intéresser aux nombres aléatoires bruts (avant le post-traitement). Dans ce cas, l'objectif est d'estimer l'entropie (et donc d'établir l'imprévisibilité du TRNG), mais également de déterminer les facteurs qui peuvent influencer la qualité de la sortie en lien avec les paramètres de la source et ceux du *design*. Le modèle peut ensuite s'intéresser aux nombres aléatoires en sortie du post-traitement arithmétique afin de valider son efficacité (le fait qu'il corrige bien les défauts de la source) et la qualité statistique des bits de sortie.

Les auteurs de [KS01] précisent que, idéalement, un modèle stochastique d'un TRNG doit proposer une famille de distributions qui décrit les nombres aléatoires en sortie du TRNG. Soient $(y_i)_{i\in\mathbb{N}}$ les valeurs que prennent les nombres aléatoires, et soient $(Y_i)_{i\in\mathbb{N}}$ les variables aléatoires qui y sont associées. Comme les mots récupérés en sortie d'un TRNG sont généralement une concaténation de n bits, les auteurs notent qu'il est naturel de s'intéresser à l'entropie conditionnelle $H(Y_{n+1}|Y_1 = y_1, Y_2 = y_2, ..., Y_n = y_n)$, ce qui correspond à une situation réelle où l'attaquant connaît une sous-séquence $\{y_1, y_2, ..., y_n\}$ de bits de sortie.

Enfin, dans le contexte de suites binaires issues d'un TRNG ou d'un DRNG, il est très important de différencier les calculs d'entropie faits sur les variables aléatoires (à travers leur modélisation) et ceux faits sur leurs réalisations (par exemple les tests d'entropie présents dans les batteries de tests statistiques). Un DRNG peut produire des suites qui passent ces tests statistiques (*i.e.* qui ont l'air aléatoires) tout en ayant une entropie par bit nulle (les bits de sortie se suivent de manière déterministe avec une probabilité égale à 1).

4.4. Les post-traitements

Les suites générées par les TRNG dans les circuits numériques souffrent souvent de défauts statistiques qui proviennent de divers facteurs : défauts de la source d'entropie, asymétries au niveau de l'extracteur, ainsi que les défauts liés à l'implantation physique telle que les irrégularités au niveau transistor (par exemple des différences entre temps de montée et de descente des signaux). Ces défauts peuvent induire des distributions biaisées et non uniformes des nombres en sortie. Il existe plusieurs algorithmes de post-traitement qui permettent de corriger ces défauts. Il faut cependant noter que ces algorithmes n'introduisent pas de vrai aléa dans les suites générées. Par contre, ils peuvent compresser les données et augmenter l'entropie par bit en combinant celle de plusieurs bits indépendants.

Les post-traitements dits arithmétiques ont pour principal but de corriger les défauts statistiques des suites générées par un TRNG. Les post-traitements dits cryptographiques ont pour objectif d'augmenter la sécurité du TRNG (en prenant le relais en cas de défaillance de la partie physique) et son débit au prix d'un coût matériel relativement important.

4.4.1. Post-traitements arithmétiques

L'objectif principal d'un post-traitement arithmétique est d'éliminer le biais statistique. Von Neumann fut l'un des premiers à s'intéresser au post-traitement de suites aléatoires imparfaites [VN51]. Le principe de son correcteur est très simple : les bits de sortie sont considérés par paires successives ; si une paire contient deux bits différents alors un bit de sortie est produit correspondant au premier bit de la paire ; si la paire contient deux bits identiques alors rien n'est produit à la sortie et la paire suivante est évaluée. Cette technique a l'avantage de réduire le biais localement en éliminant les longues suites de '1' et de '0'. Par contre, elle impose un débit de sortie variable et donc des contraintes d'implantation supplémentaires au niveau de l'interface de sortie du TRNG. D'autre part, son efficacité dépend fortement du degré de dépendance des bits d'entrée.

Les filtres de parité sont aussi très couramment utilisés pour réduire le biais statistique d'une suite aléatoire. Le principe d'un filtre de parité consiste à effectuer l'addition en base 2 de n bits successifs pour générer un bit de sortie, ce qui correspond à une opération de OU exclusif. n est appelé l'ordre du filtre. Si les bits d'entrée sont indépendants, alors plus n est grand, moins le bit de sortie est biaisé. Par ailleurs, le débit est divisé par un facteur n. E. Davis s'est intéressé à la modélisation de ce type d'algorithme. Dans [Dav02], il fournit une relation simple qui lie la probabilité du bit de sortie généré P_{out} à celles des bits d'entrée P_{in} et à l'ordre du filtre n, celle-ci est exprimée par l'équation 1.2. Comme on peut le voir, si P_{in} est différent de 0 ou 1, alors P_{out} converge vers $1/2$ quand n tend vers l'infini. L'auteur traite également le cas de bits d'entrée corrélés.

$$P_{out} = 0.5 - 2^{n-1}(P_{in} - 0.5)^n \tag{1.2}$$

Enfin, les registres à décalage à rétroaction linéaire (*Linear Feedback Shift Registers - LFSR*) peuvent également être utilisés comme post-traitements arithmétiques (par exemple dans [Gol04]). Il s'agit d'une chaine de registre avec une rétroaction de sa sortie sur un ou plusieurs de ses étages via des portes "ou exclusif" (XOR). Leur modélisation est basée sur l'étude de polynômes caractéristiques définis par la taille du LFSR et les positions des rétroactions. Cette étude permet de décrire l'évolution du système, sa périodicité et sa complexité. Cependant, il faut être prudent avec leur utilisation car leur comportement complexe peut masquer une faible entropie en entrée (ces structures peuvent générer de bonnes séquences d'un point de vue statistique sans réelle entropie en entrée). Ceci est dangereux car ces structures sont très prévisibles. Par exemple, l'algorithme de Berlekamp-Massey, introduit dès 1969 par J. Massey, permet de déterminer à la fois le polynôme caractéristique et l'initialisation des registres de LFSR à partir de séquences enregistrées de sa sortie. De manière générale, contrairement aux post-traitements cryptographiques, les post-traitement arithmétiques n'apportent aucune garantie de sécurité supplémentaire au TRNG.

4.4.2. Post-traitements cryptographiques

Les post-traitements cryptographiques apportent une garantie de sécurité supplémentaire au TRNG (la première étant l'entropie en sortie du générateur) en plus d'avoir des propriétés de diffusion qui corrigent les défauts statistiques. Il s'agit d'algorithmes cryptographiques, suffisamment

durcis d'un point de vue sécurité (selon des critères proches de ceux présentés dans la section 1.2.2 pour les DRNG cryptographiquement sûrs), et ayant une période suffisamment longue pour prendre le relai en cas d'attaque sur le coeur du générateur (source et/ou extracteur d'entropie) ou défaillance de la source d'aléa.

Ils consistent principalement en des algorithmes de chiffrement par bloc (AES, DES), des fonctions de hachage (SHA-2 ou plus récemment SHA-3 [7]) ou des fonctions résilientes. Ces primitives cryptographiques sont standardisées et durcies d'un point de vue sécurité. Leur cout matériel peut s'avérer important et parfois disproportionné par rapport à celui du TRNG, mais elles sont nécessaires dans les applications les plus critiques en terme de sécurité (par exemple la génération de signatures DSA). Enfin, remarquons qu'il est toujours possible d'utiliser un DRNG comme post-traitement cryptographique à condition que celui-ci remplisse les critères de la classe DRG. 3 décrite dans Tab. 1.1 [KS01].

4.5. Evaluation statistique des suites issues d'un TRNG

L'évaluation statistique des suites de sorties d'un TRNG diffère légèrement de la procédure utilisée pour évaluer les DRNG. La sortie du TRNG mise à disposition de l'usager (après un post-traitement arithmétique ou cryptographique) doit bien sûr être validée avec les batteries de tests standard FIPS 140-1, NIST SP 800-22, DIEHARD, etc. Cependant, les auteurs de [KS01] remarquent qu'il est raisonnable de tolérer de petits écarts statistiques au niveau des nombres aléatoires bruts d'un TRNG, en effet ceux-ci ne sont jamais "idéaux". La procédure AIS31 propose donc une série de tests qui reprend quelques uns de ceux de FIPS 140 et NIST SP 800-22 mais en adaptant les distributions de référence de ces tests et leurs seuils de passage. Ces tests sont au nombre de neuf, ils sont décrits dans Tab. 1.2. Le nombre de données minimal nécessaire pour réaliser ces tests est de l'ordre de $1Mo$. Par ailleurs, les auteurs notent que dans certains cas, il peut être judicieux de proposer un test statistique qui prenne en compte les distributions attendues en fonction de celles de la source d'aléa (comme cela est fait dans [KS08]).

Test	Description
T0	Test de disjonction (*disjointness test*)
T1	Test monobit (biais)
T2	Test de poker
T3	Test *runs*
T4	Test *long runs*
T5	Test d'auto-corrélation
T6	Test d'uniformité
T7	Test d'homogénéité
T8	Test d'entropie (équivalent à *Maurer universal test* dans NIST SP 800-22)

Table 1.2 – Liste des tests statistiques proposés dans [KS01]

Par ailleurs, la méthodologie AIS31 définit les procédures d'applications de ces tests en fonction de la sortie testée. Une première procédure est proposée pour valider les bonnes propriétés

7. http ://keccak.noekeon.org/

statistiques des suites (basée sur les tests T0 à T5). Elle peut être appliquée à la fois pour la sortie post-traitée d'un TRNG mais aussi pour les DRNG. La deuxième procédure proposée est quant à elle, spécifique à l'évaluation de la sortie brute d'un TRNG (non post-traitée). Elle a pour objectif de vérifier que l'entropie par bit est suffisamment grande (en apparence, comme pour tous les tests statistiques), des écarts statistiques tels qu'un faible biais peuvent être tolérés. Cette procédure utilise les tests T6 à T8. D'autre part, [KS01] propose un jeu de paramètres recommandés pour l'utilisation des tests NIST SP 800-22 dans le contexte des TRNG.

Enfin, il est très important de préciser que dans le cas des TRNG, l'interprétation des tests doit être en lien avec le fonctionnement interne du TRNG. Plus le fonctionnement est clair et maîtrisé, plus il est facile d'établir un lien effectif entre l'entropie réelle du générateur et les résultats des tests statistiques : cela ne veut absolument pas dire que les tests permettent de vérifier l'entropie, mais dans certains cas (si l'extraction d'entropie est suffisamment "transparente"), les résultats statistiques dépendent directement du taux d'entropie du générateur (l'inverse n'est pas vrai). D'autre part, plus ces tests sont appliqués au plus près de la source, plus il est facile d'en interpréter le résultat (par exemple, un test statistique en sortie d'un post-traitement cryptographique ne donne absolument aucune indication sur l'entropie réelle par bit).

4.6. Classes de TRNG selon AIS31 et domaines d'application

AIS31 définit les différentes classes de TRNG physiques [8] (utilisant une source d'aléa physique) en fonction de critères basés sur la sécurité, et établit le cadre d'application de ces différentes classes dans les systèmes cryptographiques. Ces classes sont détaillées dans Tab. 1.3.

Classe de RNG	Critères
PTG.1	TRNG physique, testabilité interne permettant de détecter un échec total (*total failure*) de la source d'entropie et une défaillance statistique non tolérable de la sortie (éventuellement post-traitée)
PTG.2	PTG.1 avec une modélisation stochastique de la source d'entropie et application des tests en ligne sur la sortie non post-traitée
PTG.3	PTG.2 avec un post-traitement cryptographique (TRNG hybride)

Table 1.3 – Classes de TRNG physiques définies par AIS31

La classe PTG.1 définit les critères pour des TRNG susceptibles d'être utilisés dans des applications cryptographiques où l'imprévisibilité des suites n'est pas nécessaire. La métrique de qualité de cette classe n'exclut pas que les nombres aléatoires puissent être devinés.

La classe PTG.2 définit les critères pour des TRNG utilisés pour générer des clés cryptographiques (par exemple pour le chiffrement par bloc ou pour RSA), des bits de padding aléatoires, des graines pour les DRNG ou pour tout autre application cryptographique ayant des exigences de

8. Il existe des TRNG utilisant une "prétendue" source d'aléa non-physique (par exemple les mouvements de la souris d'un ordinateur ou les temps entre frappes claviers), cependant leur sécurité est douteuse de notre point de vue

sécurité similaires (en particulier en ce qui concerne l'imprévisibilité des suites). La classe PTG.2 introduit donc un critère de modélisation stochastique pour quantifier l'entropie et établir l'imprévisibilité de la sortie du générateur. Elle reprend également les critères de PTG.1 mais elle requiert l'application de tests en lignes sur les nombres aléatoires non post-traités. Les nombres aléatoires produits devraient être pratiquement indiscernables d'une suite de nombres aléatoires indépendants et uniformément distribués (sortie d'un RNG idéal).

Enfin, à la différence de PTG.2, PTG.3 s'appuie sur deux garanties de sécurité complètement dissociables : une garantie de sécurité théorique assurée par la partie physique du générateur (entropie des nombres aléatoires internes, comme pour PTG.2), ainsi qu'une garantie de sécurité pratique assurée par le post-traitement cryptographique avec mémoire (celui-ci peut être interprété comme un DRNG de classe DRG. 3). La classe PTG.3 définit donc les exigences pour des RNG susceptibles d'être utilisés dans n'importe quelle application cryptographique, y compris les plus exigeantes en terme de sécurité. Une application typique de cette classe de RNG est la génération de clés pour les signatures DSA (Digital Signature Algorithm) et ECDSA (Ellyptic Curve Digital Signature Algorithm).

4.7. Autres critères liés au principe et à son implantation

La qualité de la source d'aléa est déterminante dans la conception d'un TRNG. Par exemple, dans le cas du bruit, celui-ci doit se rapprocher le plus possible d'un bruit blanc théorique (un spectre plat sur l'ensemble des fréquences). Or, en pratique, le bruit comporte plusieurs composantes qui peuvent parfois être déterministes). Le bruit thermique, dû aux mouvements aléatoires des porteurs de charge, est celui qui se rapproche le plus des caractéristiques d'un bruit blanc. Cependant, il n'existe actuellement pas de dispositif qui permette de capter le bruit thermique indépendamment des autres composantes de bruit présentes dans le circuit. On cherche donc à minimiser les phénomènes indésirables par un design soigné (alimentation par une tension régulée, utilisation de structure simple n'introduisant pas de bruit dépendant des données etc.).

Comme il est primordial de caractériser la source d'aléa afin de permettre la construction d'un modèle du TRNG, celle-ci doit être facilement accessible à la mesure. Cependant, il faut faire attention à ne pas utiliser de source externe car cela induit une faille de sécurité importante et dont il est difficile de se protéger : elle serait facile d'accès à un attaquant et donc manipulable. La source d'aléa doit donc provenir de l'intérieur des circuits. Le *jitter* a l'avantage d'être facilement mesurable et caractérisable car il nécessite uniquement l'accès au signal numérique qui le véhicule (que cela soit à l'intérieur ou à l'extérieur de la puce via une broche de sortie). Dans certains TRNG, il est difficile d'accéder à la source d'aléa et de la mesurer (par exemple des TRNG basés sur l'initialisation de mémoires SRAM [HBF07]).

La structure et le fonctionnement de l'extracteur d'entropie conditionnent grandement la construction d'un modèle stochastique du TRNG. Idéalement, il doit être séparé de la source d'aléa d'un point de vue architectural et ne pas influer sur cette dernière. Son fonctionnement doit être complètement maîtrisé, si possible simple et ne générant pas de pseudo-aléa. Les TRNG où la source d'aléa est imbriquée dans le mécanisme d'extraction ne permettent généralement pas de

modéliser efficacement l'entropie à leur sortie ([Gol06], [Tka03]).

En ce qui concerne la surface et la consommation des TRNG, il est intéressant de noter que dans plusieurs cas, le taux d'entropie extrait dépend directement de la surface occupée par le TRNG (par exemple le nombre d'oscillateurs dans [SMS07] et et le nombre de cellules de base dans [VD10]). Pour ces raisons, il est difficile de comparer des TRNG en terme de surface et consommation car les taux réels d'entropie peuvent être sensiblement différents. Néanmoins, et de manière générale pour les TRNG, la consommation n'est pas significative car elle peut être réduite par la mise en arrêt du générateur (en effet il existe très peu d'applications nécessitant un flot continu de nombres aléatoires).

Le débit théorique qu'un TRNG peut atteindre dépend des caractéristiques de la source d'aléa (spectre de fréquences, niveau de bruit, etc.) ainsi que de la technique d'extraction d'entropie (échantillonnage, comptage etc.). Dans la plupart des applications cryptographiques, le débit est un paramètre secondaire (des taux allant de 100 Kb/s à 1 Mb/s sont généralement suffisants [Fis12]). Néanmoins, comme évoqué précédemment, la plupart des post-traitements arithmétiques améliorent les caractéristiques statistiques des nombres aléatoires bruts et augmentent l'entropie par bit en compressant l'information moyennant une perte de débit qui est fonction des défauts initiaux de la suite. Dans certains générateurs, la qualité des suites générées dépend directement de la taille/nombre de blocs de base du TRNG (par exemple le nombre d'oscillateurs dans [SMS07]). Dans ce cas, si le débit initial est important, il est possible de diminuer la taille du design quitte à baisser son débit via un réglage adéquat du post-traitement arithmétique (tout en faisant attention à conserver un taux minimal d'entropie en fonction de l'application).

Enfin, les possibilités d'automatisation sont un facteur important à prendre en compte pour une exploitation industrielle d'un *design* de TRNG, on peut distinguer trois situations (la première étant la plus bénéfique, la deuxième reste acceptable) :

- L'implantation est directe et peut être adaptée à toutes les technologies (par exemple nécessitant uniquement un code VHDL), c'est partiellement le cas pour [SMS07] et [Gol06].
- L'implantation nécessite une intervention au niveau du placement/routage qui est spécifique à une famille de cartes donnée, ou bien utilise un composant spécifique à une famille de cartes donnée ([FD02]).
- L'implantation nécessite une intervention manuelle du *designer* pour chaque carte (TRNG proposé dans [KG04]), une exploitation industrielle de la version actuelle n'est pas envisageable.

4.8. Exemples de principes de TRNG selon leur niveau de sécurité

L'annexe A propose un inventaire succinct de certains TRNG cités dans la littérature avec des descriptions détaillées de chaque *design*. Dans cette section, nous proposons d'analyser trois exemples, choisis en fonction du niveau de sécurité qu'ils permettent d'atteindre : un TRNG permettant la quantification et la maîtrise de l'entropie à l'aide des paramètres du *design*, un TRNG non testable (on ne peut pas connaître son entropie réelle) et enfin un TRNG vulnérable à cause

de diverses failles de sécurité.

4.8.1. TRNG permettant de quantifier et contrôler l'entropie

Dans [FD02], V. Fischer *et al.* reprennent le schéma constitué d'une bascule et deux signaux oscillants mais introduisent une idée nouvelle : en choisissant judicieusement le rapport entre les fréquences des deux signaux oscillants, il est possible d'effectuer un balayage du signal échantillonné avec une résolution temporelle suffisamment fine pour détecter le *jitter*. De plus, il n'est pas nécessaire que l'un des signaux ait une fréquence basse. Historiquement, c'est l'une des premières implantations de TRNG destinée à des applications cryptographiques dans une cible reprogrammable. Le principe est proche des techniques d'échantillonnage cohérent, utilisées pour augmenter la résolution spectrale des FFT (*Fast Fourrier Transform*).

Si la résolution de balayage $\triangle t$ (déterminée par le rapport entre les fréquences des signaux s et clk) est de plus plus petite que les variations dues au *jitter*, alors on est sûr que certaines valeurs échantillonnées seront influencées par le *jitter* (lorsque l'échantillonnage a lieu dans la zone d'influence du *jitter* comme décrit dans Fig. 1.11). En connaissant précisément le rapport entre les fréquences, il est possible alors de déterminer la position de ces bits aléatoires afin de les extraire avec un décimateur.

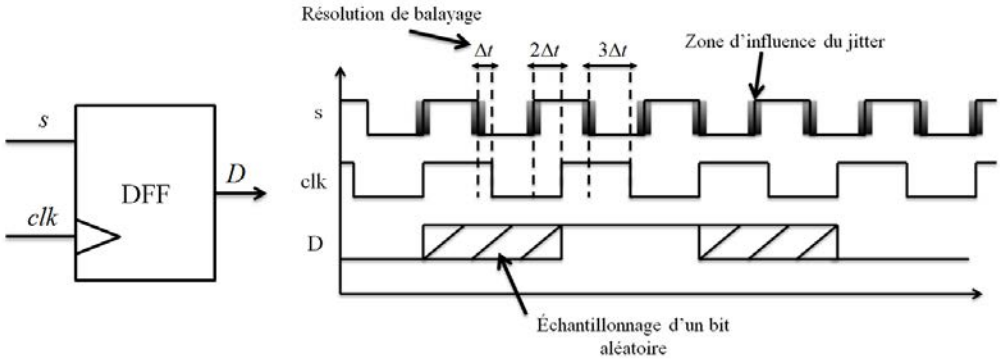


Figure 1.11 – Principe de l'échantillonnage cohérent avec une bascule D (*flip-flop*)

Pour effectuer le réglage de la résolution de balayage $\triangle t$, V. Fischer *et .al* proposent d'utiliser une PLL (*Phase-Locked Loop*). L'architecture du TRNG est donnée dans Fig. 1.12. Le signal d'horloge clk de fréquence F_{clk} passe à travers une PLL pour fournir un signal clj de fréquence F_{clj} avec :

$$F_{clj} = \frac{K_M}{K_D} F_{clk},$$

où K_M et K_D sont les paramètres de la PLL. En choisissant ces deux paramètres tels que $pgcd(K_M, K_D) = 1$ ($pgcd$ étant le plus grand diviseur commun), alors les auteurs montrent qu'on peut atteindre une résolution temporelle minimale exprimée par la relation :

$$\triangle t = \frac{pgcd(2K_M, K_D)}{4K_M F_{clk}}$$

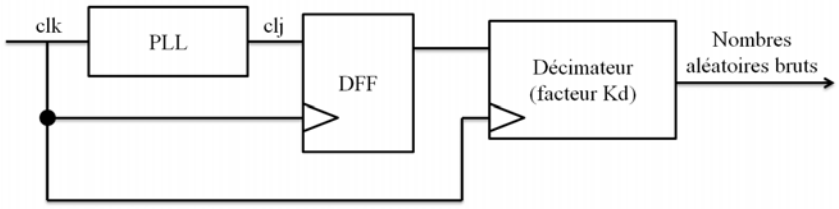


Figure 1.12 – Architecture du TRNG proposé dans [FD02]

Le principe du TRNG consiste alors à régler $\triangle t$ de manière à ce que $\triangle t < \sigma_{jit}$ où σ_{jit} est l'écart-type de la variation temporelle autour d'un front du signal échantillonné (clj). Ainsi, tous les K_D bits échantillonnés, au moins l'un d'eux aura une valeur aléatoire. Le décimateur permet d'extraire ces nombres aléatoires en sommant (XOR) K_D bits successifs à la fréquence F_{clk}. L'article présente également une analyse mathématique du principe d'extraction de *jitter* mais ne propose pas d'estimation d'entropie. Par ailleurs, les auteurs notent que dans certains cas il n'est pas possible de satisfaire la condition $\triangle t < \sigma_{jit}$, ils recommandent alors l'utilisation d'une ligne de retard afin d'augmenter les chances que l'échantillonnage recouvre la période du signal clj de manière plus fine. Une modélisation stochastique de ce TRNG est par la suite proposée par F. Bernard *et. al.* dans [BFV10]. Bien que ce modèle soit assez simplifié et qu'il ne prenne pas compte d'éventuelles dépendances entre les bits, il permet de donner une estimation du biais et de l'entropie en sortie du générateur en fonction des paramètres de la PLL (K_M et K_D), de la taille du jitter (σ_{jit}) et de la fréquence d'horloge F_{clk}. En pratique, la validité du modèle dépend fortement de la configuration de la PLL.

Les auteurs proposent des implantations du TRNG dans une cible Altera NIOS APEX EP20K200-2X, la résolution temporelle atteinte est de 7.2 ps. Par la suite, les auteurs proposent dans [FDSB04] plusieurs autres implantations dans une carte DSP (*Digital Signal Processing*) Altera Startix, dont une configuration utilisant deux EPLL (*Enhanced Phase-Locked Loop*) permettant des débits atteignant 1 Mb/s.

L'approche proposée dans [FD02] présente deux points forts qu'il est important de souligner : 1) la source d'aléa est caractérisée et mesurée, 2) le réglage de l'entropie s'effectue en fonction de cette mesure ($\triangle t$ est choisi en fonction de σ_{jit}) grâce aux paramètres de l'extracteur (PLL). Ce principe permet donc d'appliquer facilement les recommandations de la section 4.2, notamment en ce qui concerne la testabilité interne et le *monitoring* de l'entropie. Cependant, son point faible reste le coût matériel d'une PLL (bien qu'une PLL dédiée soit maintenant disponible dans la plupart des FPGA récents).

4.8.2. TRNG non testable

Dans [Gol04] et [Gol06], Golic *et al.* proposent une architecture de TRNG hybride où le mécanisme d'extraction d'entropie est profondément imbriqué dans un automate déterministe. La source d'aléa utilisée est le *jitter*, qui influe sur les états internes d'automates déterministes basés sur deux oscillateurs asynchrones mélangeant le fonctionnement d'oscillateurs en anneau à inver-

seurs et celui de LFSR : FIRO (*Fibonacci Inverter Ring Oscillator*) et GARO (*Gallois Inverter Ring Oscillator*). La structure de GARO et FIRO est donnée dans Fig. 1.13. Chaque oscillateur est décrit par son polynôme caractéristique, son degré déterminant la taille de la structure et ses coefficients les boucles de rétro-action dans chaque étage. Par exemple, un GARO de polynôme caractéristique $x^5 + x^3 + 1$ sera composé de 6 étages et aura une boucle de rétro-action dans son sixième, quatrième et premier étage. En théorie, le signal produit à la sortie de chacune de ces structures est un signal numérique contenant des séries de '1' et de '0' avec des durées pseudo aléatoires, celles-ci étant influencées par le *jitter*. En effet, l'utilisation d'éléments asynchrones sans aucun contrôle sur le séquencement des opérations fait qu'il y a souvent des événements qui occurrent avec des délais très courts dans certains noeuds de la structure. C'est précisément à ce niveau que se passe l'extraction d'entropie : lorsque des transitions électriques sont proches dans le temps dans un noeud de la structure, le *jitter* influe sur l'état interne de l'automate (ainsi que les suivants). Les auteurs montrent cela dans la pratique en réinitialisant les oscillateurs et en vérifiant que les états internes varient en fonction du *jitter*.

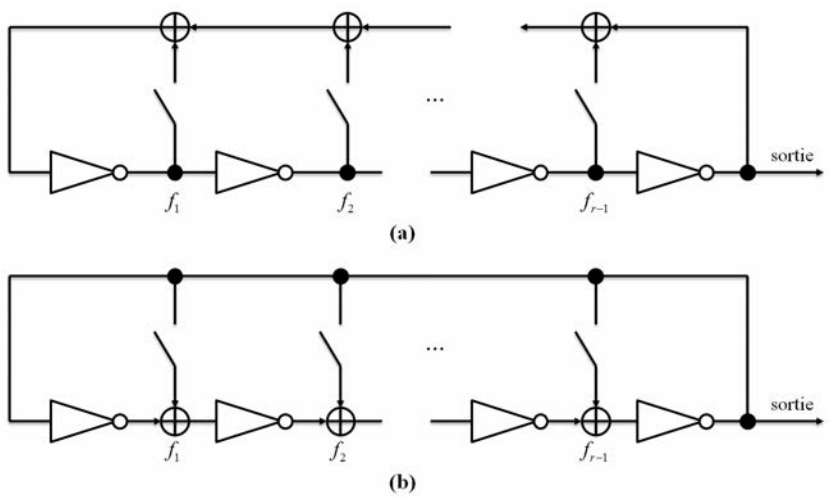


Figure 1.13 – (a) Architecture de l'oscillateur FIRO (b) Architecture de l'oscillateur GARO

L'architecture du générateur proposée dans [Gol04] est donnée dans Fig. 1.14. Les deux signaux en sortie de FIRO et GARO sont combinés à l'aide d'un OU exclusif (XOR). Les auteurs ne justifient pas clairement dans le papier la pertinence d'un tel choix. Le signal en sortie du XOR est échantillonné à fréquence constante pour produire les nombres aléatoires bruts. Ces nombres sont ensuite post-traitées avec une LFSR pour corriger leurs défauts statistiques. Du point de vue théorique, l'analyse proposée dans [Gol06] se cantonne à la partie déterministe du générateur : les auteurs étudient et déterminent par exemple les conditions sur les polynômes pour qu'il n'y ait pas de blocage du générateur (i.e. que l'automate reste un temps indéterminé dans le même état). L'extraction d'entropie n'est par ailleurs pas modélisée dans le papier. D'après les auteurs, les séquences générées en pratique présentaient un biais étonnamment important, le post-traitement utilisé est d'ailleurs assez important et consiste en une LFSR. Enfin, dans [DG07], M. Dichtl et

Y. Golic remarquent que l'oscillateur GARO a un comportement beaucoup plus chaotique que l'oscillateur FIRO (dans ce dernier les temps entre événements sont plus longs). Ils introduisent deux améliorations au générateur : seul l'oscillateur de type GARO est utilisé, il est réinitialisé à chaque bit produit pour limiter les problèmes de dépendance, et une bascule de type *toggle* est introduite à la sortie de l'oscillateur GARO (ceci permettrait, d'après les auteurs, de doubler l'entropie théorique en sortie).

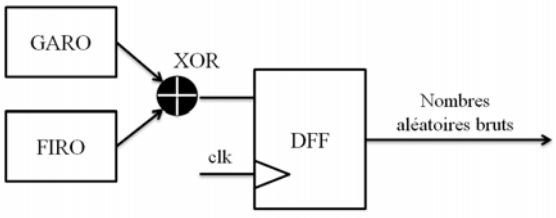


Figure 1.14 – Architecture du TRNG proposé dans [Gol04]

Bien qu'il semble évident que le fonctionnement des oscillateurs FIRO et GARO soit fortement influencé par le *jitter*, le mécanisme d'extraction d'entropie reste difficilement modélisable et dissociable de la fonction déterministe, ce qui rend l'estimation de l'entropie réelle en sortie particulièrement ardue comme cela est le cas dans [Tka03] : ce TRNG n'est donc pas testable.

4.8.3. Exemple d'un TRNG vulnérable : le RO-TRNG

L'architecture du RO-TRNG (étudiée dans [SMS07] et [WT08]) est décrite dans Fig. 1.15. Les sorties de L oscillateurs en anneau, notées $(C_i)_{1\leq i\leq L}$, sont combinées à l'aide d'un arbre de XOR, *a priori* elles ont des fréquences différentes (au mieux très proches). Le signal résultant ψ est échantillonné avec une horloge *clk*. Chaque événement survenant au sein de n'importe lequel des signaux $(C_i)_{1\leq i\leq L}$ est transmis via le XOR dans le signal ψ. Plus le nombre de ces signaux est important, plus il y a de chances que l'échantillonnage survienne souvent dans une zone influencée par le *jitter*. Ce principe théorique permet de s'affranchir de l'accumulation du *jitter* et laisse donc envisager des débits très importants. On peut noter qu'en pratique, cette structure génère nécessairement du pseudo-aléa dû à la déviation de phase entre les oscillateurs car ceux-ci ne sont pas synchronisés.

D'un point de vue théorique, les auteurs proposent deux modèles pour décrire le fonctionnement du générateur. Le premier modèle, qui est probabiliste, a pour but de déterminer le nombre d'oscillateurs nécessaire pour réaliser une répartition uniforme et suffisamment fine des événements dans le signal ψ. Le second modèle est un modèle stochastique, il détermine l'entropie en sortie du générateur en fonction des paramètres du générateur. Utilisés conjointement, ces deux modèles permettent de dimensionner le TRNG en fonction des paramètres du *jitter*. Celui-ci est modélisé par une loi normale caractérisée par son écart-type σ. L'approche proposée peut être résumée en ces différentes étapes : **1)** Dans un premier temps, la période d'oscillation T (supposée la même pour tous les oscillateurs) est divisée en N intervalles égaux ayant une durée de l'ordre de celle

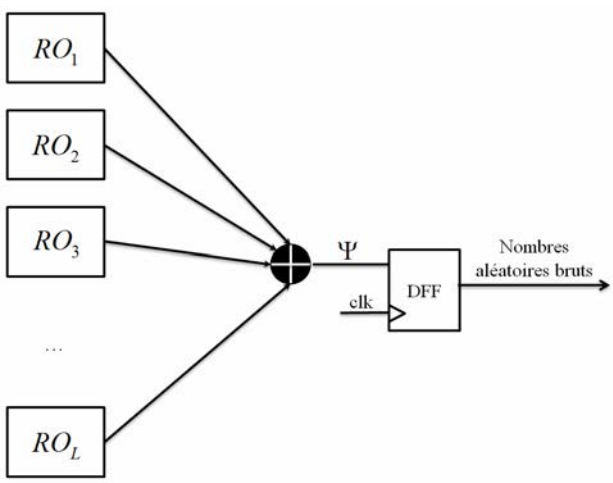


Figure 1.15 – Architecture du TRNG proposé dans [SMS07]

des variations du *jitter*, ces intervalles sont appelés "urnes". La durée de ces urnes est calculée en fonction de l'écart-type du *jitter* σ et du taux d'entropie désiré à l'aide d'un modèle stochastique. **2)** Etant donné que le concepteur n'a aucun contrôle sur les phases des oscillateurs, le problème est le suivant : combien faut-il en utiliser afin de statistiquement remplir chacune de ces urnes avec au moins un événement avec une probabilité importante. Pour déterminer ce nombre, les auteurs reprennent le problème probabiliste bien connu du collecteur de coupons dont l'exemple le plus simple est le suivant : un personnage lance des ballons dans N paniers avec une probabilité égale de remplir un panier ou un autre, combien faudrait-t-il lancer de ballon pour que chaque panier en contienne au moins un ? D'après la résolution de ce problème, ce nombre peut être estimé par $Nlog(N)$. **3)** Comme le nombre d'oscillateurs nécessaire pour le remplissage augmente de manière exponentielle avec le nombre d'urnes, celui-ci peut vite devenir un facteur limitant pour une réalisation pratique du générateur. Pour remédier à cela, les auteurs proposer de réduire ce nombre d'oscillateurs en utilisant une fonction résiliente qui rehausse l'entropie en sortie du générateur et augmente sa tolérance aux attaques et aux fautes.

Pour illustrer la méthode, les auteurs proposent un exemple concret sur la base des paramètres suivants : les oscillateurs sont constitués de 13 étages, ils ont une période de 25 ns et l'écart-type du *jitter* est de 0.5 ns (valeur largement surestimée de notre point de vue). Ils estiment alors qu'il faut utiliser 114 oscillateurs et un code BCD (*Binary Coded Decimal*) de paramètres [256,16,113] afin d'obtenir une entropie de 0.97 par bit de sortie.

Il est assez aisé de vérifier que ce type de TRNG est actuellement l'un des plus étudiés et des plus utilisés. Son attractivité vient de plusieurs facteurs : son implantation est particulièrement aisée dans toutes les cibles (ASIC et FPGA) et ne requiert pas d'intervention manuelle du concepteur (un code VHDL est généralement suffisant), les suites générées ont généralement une très bonne qualité statistique et ne nécessitent pas toujours un post-traitement, et enfin les débits obtenus sont très importants (de l'ordre de la centaine de Mbit/s). Du fait de sa popularité, ce générateur est

aussi l'un des plus étudiés et commentés par la communauté scientifique. Plusieurs travaux récents mettent en évidence les failles de sécurité importantes liées à la fois au principe du générateur et à son implantation.

En réalité le modèle se base sur une hypothèse probabiliste qui ne peut être systématiquement garantie (celle stipulant que "si suffisamment d'oscillateurs sont utilisés alors quelque soit le moment d'échantillonnage on est sûr d'échantillonner près d'un front") et sur d'autres hypothèses qui ne sont pas tenables dans la réalité (indépendance des oscillateurs, égalité des fréquences, amplitude du *jitter* ...). Ensuite, l'architecture présente plusieurs problèmes.

L'un des premiers défauts dans l'architecture originale fut relevé par M. Dichtl qui remarque que l'arbre de XOR ne peut pas correctement restituer le signal de sortie au vu du nombre et de la haute fréquence des signaux issus des oscillateurs. Wold et Tan apportent rapidement une solution à ce problème dans [WT08] en insérant des bascules à la sortie de chaque oscillateur comme le montre Fig. 1.16. D'un point de vue théorique, ceci ne change en rien le principe initial. En pratique, les auteurs ont proposé dans l'article une implantation du générateur n'utilisant que 50 anneaux à inverseurs à 3 éléments et passant l'ensemble des tests statistiques standards avec un débit de 100 Mbit/s. Là encore, les auteurs ne portent aucun intérêt à l'entropie du générateur. Ceci met donc en évidence un autre problème plus grave dans ce générateur : le vrai aléa est largement masqué par le pseudo-aléa dû aux différences de fréquences des oscillateurs. En effet ceux-ci ne peuvent avoir des fréquences égales (différences de routages, variabilité des procédés de fabrication, etc.). Ceci introduit donc une dérive de phase (*phase drift*) qui apparaît comme des variations pseudo-aléatoires dans les durées des niveaux hauts et bas du signal combiné.

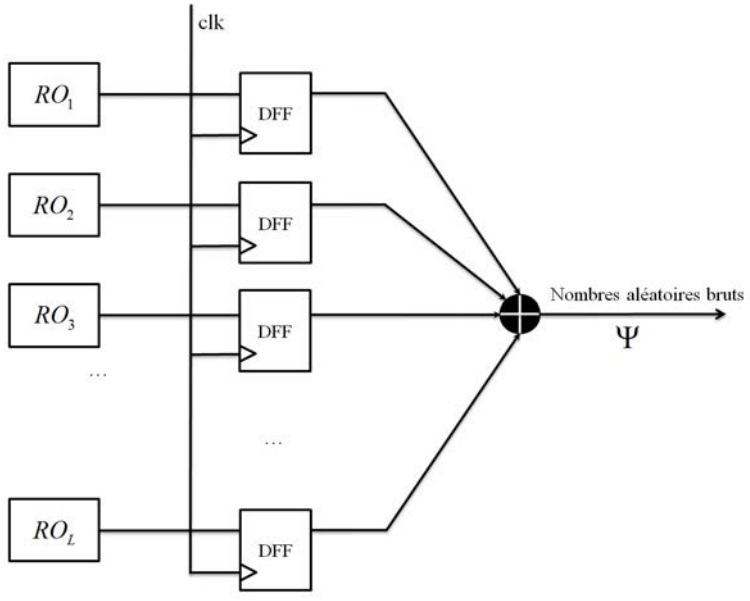


Figure 1.16 – Amélioration proposée par Wold et Tan dans [WT08]

Pour évaluer l'ampleur du comportement déterministe de ce générateur, V. Fischer *et al.* en

ont réalisé dans [BBFV10] un modèle VHDL haut-niveau en intégrant de légères variations entre les fréquences des oscillateurs (représentant la variabilité de procédé de fabrication, et différences de délais dues au routage). Les auteurs ont constaté qu'un design composé seulement de 18 oscillateurs anneaux produisait des suites qui passaient les tests statistiques NIST SP 800-22 (qui sont parmi les tests les plus sélectifs), et cela sans introduire de source d'aléa (*jitter*) dans le modèle. Ceci implique que le concepteur doit être extrêmement prudent en utilisant ce TRNG : son comportement statistique est loin de refléter le taux réel d'entropie issu de la vrai source d'aléa (*jitter*).

Les mêmes auteurs ont ensuite étudié le comportent d'oscillateurs en anneau à inverseurs implantés dans des circuits FPGA, et ont montré que dans des conditions réelles, il apparaissait de fortes dépendances entre ces oscillateurs. La manifestation la plus marquante (et dangereuse pour le comportement du générateur) observée fut un état de synchronisme total ou partiel entre deux oscillateurs ou plus, durant lequel ces oscillateurs basculaient leurs signaux électriques exactement au même moment, réduisant de manière drastique l'entropie en sortie du générateur. Ce phénomène a été appelé *locking* en référence au verrouillage des signaux sur les mêmes temps de basculement. En pratique, les auteurs montrent que jusqu'à 25% des oscillateurs implantés dans une même cible Altera Cyclone 3 peuvent être totalement ou partiellement "verrouillés", ce qui a des conséquences dramatique pour l'entropie en sortie du générateur.

Ce phénomène de *locking* constitue le point d'entrée des attaques électromagnétiques de type harmonique. Celle-ci cherchent à provoquer le *locking* d'un ou plusieurs oscillateurs sur la fréquence de rayonnement pour réduire l'entropie et pouvoir prédire les bits de sortie avec une probabilité non négligeable. Les travaux présentés dans [BBA+12] démontrent la faisabilité, la pertinence d'une telle attaque et les dangers potentiels qu'elle représente.

Pour conclure sur ce TRNG, le principe théorique proposé est intéressant, mais sa réalisation présente des défauts importants et ne permet pas d'exploiter tout le potentiel du principe. A l'instar des générateurs hybrides, le comportement déterministe causé par le *phase drift* des oscillateurs introduit trop de pseudo-aléa pour pouvoir évaluer l'efficacité de l'extraction d'entropie (il peut également donner un "faux sentiment" de sécurité). Le modèle proposé pour l'estimation d'entropie est d'ailleurs faible car il repose sur des hypothèses difficilement vérifiables. Par ailleurs, la susceptibilité des oscillateurs aux phénomènes de *locking* constitue une menace réelle pour ce type de générateur. Ceci est particulièrement vrai à la vue du développement récent des attaques actives par rayonnement électromagnétique ([BBA+12], [BBAF13]).

5. Conclusion

Le principal critère de sécurité d'un TRNG doit toujours être l'imprévisibilité de sa sortie. En effet, il ne sert à rien d'avoir recours à un TRNG si on cherche uniquement la qualité statistique des suites (un DRNG peut remplir cette fonction tout en étant plus rapide et plus facile à implanter). Par ailleurs l'imprévisibilité de la sortie d'un TRNG, est dans l'idéal, directement liée à l'imprévisibilité de sa source d'entropie. On peut faire ici l'analogie avec le principe de Kerckhoffs pour les algorithmes de chiffrement : un *design* de TRNG doit être parfaitement connu et maîtrisé, sa

sécurité étant complètement basée sur l'imprévisibilité de sa source d'entropie. Très peu de TRNG proposés dans la littérature respectent ce principe. Dans beaucoup de cas, l'extraction d'entropie repose sur des principes complexes et pas suffisamment maîtrisés ([Gol04], [HBF07]) et le comportement statistique ne reflète pas du tout l'entropie réelle du générateur ([Tka03], [SMS07]). On peut cependant citer [FMC85], [FD02] et [KS08] qui ont une architecture suffisamment claire et maîtrisée pour permettre une réelle estimation de l'entropie de leur sortie (et donc d'établir l'imprévisibilité des suites qu'ils génèrent).

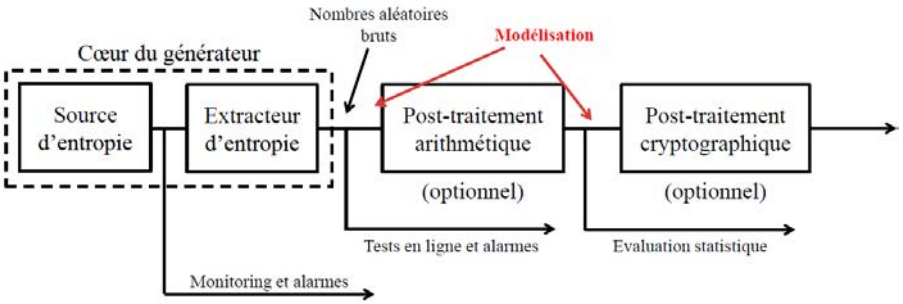


Figure 1.17 – Schéma de conception et d'évaluation de TRNG avec la sécurité renforcée

Fig. 1.17 résume les principaux critères modernes pour la conception et l'évaluation de TRNG (dans des circuits numériques en particulier). La source d'aléa doit être parfaitement connue, caractérisée et si possible séparée d'un point de vue architectural de l'extracteur d'entropie. Celui-ci doit avoir un fonctionnement suffisamment clair pour permettre une modélisation effective des nombres aléatoires à sa sortie. L'objectif de ce modèle est d'établir l'imprévisibilité effective de la sortie du TRNG en calculant l'entropie en lien avec des mesures sur la source d'aléa. Comme les sources physiques ne sont jamais parfaites, des défauts statistiques à ce niveau peuvent être tolérés. Ils sont corrigés à l'aide d'un post-traitement arithmétique. L'objectif du modèle en sortie de ce post-traitement est de montrer qu'il effectue bien son rôle, *i.e.* qu'il corrige effectivement les défauts de la source. L'évaluation statistique doit obligatoirement se faire sur les nombres en sortie du post-traitement arithmétique, et de préférence aussi sur les nombres non post-traités bien que les critères ne soient pas les mêmes (AIS31 définit par exemple la procédure pour appliquer ses tests en fonction de la sortie testée). Des tests adaptés à la source d'aléa (qui prennent en compte ses caractéristiques et défauts) peuvent être envisagés pour les nombres non post-traités. Enfin, le post-traitement cryptographique peut être considéré comme un deuxième bloc complètement séparé du point de vue de la conception et l'évaluation. Son rôle principal est d'apporter une seconde garantie de sécurité à l'application qui utilise les séquences (la première étant la partie physique du générateur) : il utilise des graines issues du TRNG durant son fonctionnement normal et prend le relais en cas de défaillance de la partie physique du générateur (par exemple une attaque physique sur la source d'aléa).

Du point de vue sécurité, plusieurs niveaux peuvent être envisagés en fonction de l'application visée. Dans le cas d'implantation au sein d'un système cryptographique, les moyens de *monitoring* des bits internes et alarmes doivent être proposés afin de détecter des défaillances graves. Dans les

applications sensibles telles que la génération de clés pour la cryptographie symétrique, l'utilisation d'un post-traitement cryptographique est requise comme garantie de sécurité supplémentaire. Enfin pour obtenir des niveaux de sécurité encore plus importants et une meilleure robustesse aux attaques, et si le principe de TRNG le permet, les moyens de *monitoring*, tests et alarmes peuvent être appliqués directement sur la source d'entropie.

Chapitre 2.

Les oscillateurs en anneau auto-séquencés

Sommaire

1. Introduction

Les oscillateurs en anneau à inverseurs (*Inverter Ring Oscillators - IRO*) sont des structures oscillantes simples composées de chaines d'inverseurs dans lesquelles une transition électrique (ou "événement") se propage de manière périodique. Comme nous le constations dans le chapitre précédent, un nombre important de TRNG utilisent le *jitter* d'un ou plusieurs anneaux à inverseurs comme source d'aléa. Il faut aussi savoir que, malgré leur faible précision en fréquence, ces oscillateurs sont largement utilisés dans les systèmes numériques pour la génération d'horloges locales et de références fréquentielles. Leur attrait vient de leur faible coût et de leur très bonne intégration dans le flot de conception CMOS standard. Cependant, ils souffrent d'une forte dépendance aux conditions environnantes et sont très sensibles à la variabilité des procédés de fabrication [YKBS10]. Outre leur très faible précision (particulièrement dans les hautes fréquences), ils ne sont pas adaptés pour générer des signaux multi-phasés : leur résolution temporelle est limitée par les délais de propagation d'un étage. Par ailleurs, leur structure n'offre pas de possibilités de configuration dynamique numérique : pour modifier leur fréquence sans passer par un réglage analogique, il faut changer leur nombre d'étages.

Ces dernières années ont vu l'émergence d'un nouveau type d'oscillateurs tirant parti des techniques de conception asynchrone, appelés oscillateurs en anneau auto-séquencés (*Self-timed Ring Oscillators - STR*). Ces oscillateurs utilisent un protocole de requêtes et acquittements pour ordonnancer les transitions électriques au sein de leur structure. Ainsi, plusieurs transitions électriques peuvent se propager simultanément dans l'anneau sans risque de collisions. Cette caractéristique, couplée à des effets analogiques propres à la structure des étages, donne lieu à un comportement temporel très remarquable : dans certaines conditions (qui sont faciles à contrôler), les événements se propagent de manière régulière avec un contrôle intrinsèque des temps qui les séparent (d'où l'appellation "anneaux auto-séquencés").

Les anneaux auto-séquencés sont naturellement reconfigurables : leur fréquence ne dépend pas directement du nombre d'étages, mais plutôt du nombre d'événements initialisés. Ils permettent la génération de signaux avec des résolutions temporelles extrêmement fines, celles-ci ne sont pas limitées par les délais de propagation des étages [Fai09]. Enfin, les travaux de J. Hamon on mis en évidence leur bonne robustesse aux variations de tension, à la température et aux procédés de fabrication [HFMR08]. Ces trois caractéristiques semblent particulièrement intéressantes dans l'optique d'améliorer la robustesse et les performances des TRNG actuels à base d'anneaux à inverseurs. Dans le chapitre 3 de ce document, nous présentons un nouveau principe de TRNG qui tire parti de ces trois caractéristiques.

Ce chapitre présente les oscillateurs en anneaux auto-séquencés, leur comportement temporel fin et les modèles utilisés pour les décrire. Dans un premier temps, la section 2 introduit les notions de base du style de conception asynchrone et les propriétés des circuits asynchrones. La section 3 présente l'architecture et fonctionnement général de deux types d'oscillateurs asynchrones : les oscillateurs en anneau à base d'inverseurs et les oscillateurs en anneau auto-séquencés. La section 4 détaille les modèles utilisés pour décrire les anneaux auto-séquencés. Enfin, la section 5 conclue ce chapitre.

2. Notions de base sur la conception asynchrone

Cette section introduit les notions de base de la logique asynchrone et décrit les propriétés fondamentales des circuits asynchrones. Tous les concepts présentés ici sont largement documentés dans le rapport de Marc Renaudin du laboratoire TIMA qui est un état de l'art exhaustif sur la logique asynchrone [Ren10]. L'annexe B est associée à cette section, elle décrit les protocoles de communication et types de codage des données permettant la réalisation de circuits asynchrones fonctionnels. Elle présente également les classes de circuits asynchrones et les avantages liés à leur utilisation.

2.1. Historique

Les circuits asynchrones représentent une classe de circuits numériques dont l'ordonnancement et contrôle sont assurés par des mécanismes internes ne nécessitant pas de recours à une horloge globale comme c'est le cas pour les circuits synchrones. La distinction entre circuits synchrones et asynchrones n'existait pas au début de la conception numérique. Les concepteurs ont commencé à privilégier le style de conception synchrone car celui-ci est plus simple à mettre oeuvre, et plus facile à intégrer dans des flots de conception standardisés.

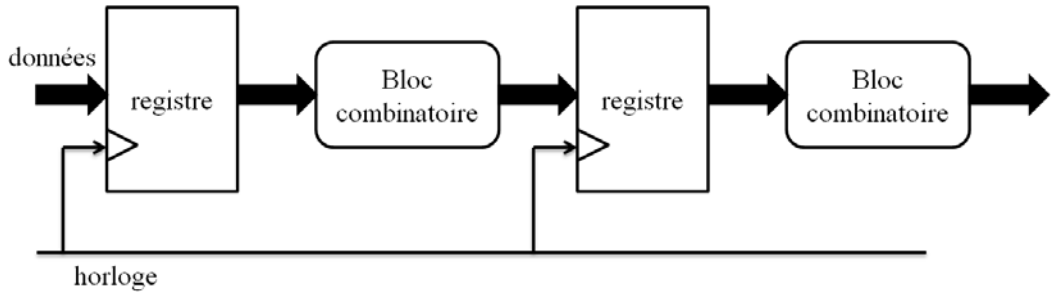


Figure 2.1 – Structure générale d'un circuit numérique synchrone

Les circuits synchrones présentent tous la même structure de base, composée de registres de mémorisation intercalés entre des blocs de logique combinatoire tel que représenté dans Fig. 2.1. Une horloge globale assure le séquencement des différents traitements effectués dans les blocs combinatoires. Ce modèle, actuellement utilisé par la plupart des concepteurs et outils de CAO modernes, est conforme au formalisme de spécification RTL (*Register transfer level*). Les circuits synchrones ont l'avantage de simplifier la conception des blocs combinatoires : celle-ci n'a plus besoin de prendre en compte les aléas logiques et effets transitoires puisque les traitements s'effectuent entre deux fronts de l'horloge (hypothèse de discrétisation temporelle). Néanmoins, leurs performances peuvent être limitées par une contrainte temporelle importante sur la période de l'horloge : celle-ci doit s'adapter au bloc combinatoire le plus lent du circuit (approche pire cas).

Actuellement, les récents progrès de la microélectronique (complexité croissante des circuits, réduction des dimensions des transistors, etc) remettent en question la pertinence du paradigme du

"tout synchrone". Les contraintes temporelles liées à l'horloge sont de plus en plus difficiles à maîtriser et deviennent rapidement un facteur limitant en terme de performances. Parmi les défis actuels que rencontrent les concepteurs, on peut citer les problèmes de distribution d'horloge, de rayonnement électromagnétique et de modularité. De ce fait, le style de conception asynchrone montre un regain d'intérêt car il semble apporter une solution naturelle à ces problèmes majoritairement liés à l'utilisation d'un signal global de synchronisation.

Les premiers tentatives de formalisation des circuits asynchrones datent des années 50 sous l'initiative de D. E. Muller et W. S. Bartky de l'université de l'Illinois [MB556]. D. E. Muller a notamment proposé un protocole de communication qui associe un signal de validité aux données (ce protocole utilise deux rails et trois états pour le codage des données), ce qui fut l'une des premières étapes de formalisme des circuits asynchrones. Ces études font écho aux travaux de D. A. Huffman sur les machines d'états asynchrones quelques années auparavant [Huf54].

Ensuite, l'un des événements marquants dans l'histoire de la conception asynchrone fut le lancement du projet *The macromodule project* par W. A. Clark en 1966 de l'université de Washington à Saint Louis [Cla67]. Ce projet a démontré les avantages de la conception asynchrone en terme de modularité permettant la conception de machines spécialisées par simple assemblage de blocs de base. L'étape suivante fut l'utilisation de "m-net" par C. L. Seitz dans les années 70 [Sei70], introduisant un formalisme proche des réseaux de Petri, qui a donné lieu à la conception du premier calculateur de type "flot de données" fonctionnel par A. L. Davis [Dav78].

Enfin, le célèbre article d' I. Sutherland *"Micropipelines"* [Sut89], paru en 1989, a largement contribué à l'intérêt de la communauté scientifique au style de conception asynchrone. Cette période voit également l'apparition du premier microprocesseur asynchrone, le *Caltech Asynchronous Microprocessor* conçu à Caltech par l'équipe d'Alain Martin en 1989 et possédant une architecture RISC. Par la suite, plusieurs processeurs asynchrones voient le jour : AMULET, MiniMIPS de Caltech, TITAC [TKI+97] ou encore Aspro (*Asynchronous processor*, conçu en 2001 au laboratoire TIMA de Grenoble).

L'exploitation industrielle des technologies asynchrones prend son essor à la fin des années 90. *Philips* fait figure de précurseur dans le domaine, la société utilise des microcontrôleurs asynchrones dès 1998 dans plusieurs de ses systèmes dont des pagers et cartes à puce. *Intel*, *IBM* ainsi que *Sun Microsystems* participent également à cet engouement : *Intel* conduit des recherches pour créer un décodeur pour le jeu d'instructions x86 entre 1995 et 1999 tandis que *Sun* inventent plusieurs techniques, les plus importantes concernent les pipelines de type GasP (Globally asynchronous, locally Synchronous). Actuellement, de nombreuses start-ups participent à la recherche et développement des circuits asynchrones : *Theseus Research Incorporated*, *Fulcrum* (rachetée par *Intel*), *Tiempo* (qui a notamment conçu des outils de CAO pour la conception asynchrone qui s'intègrent dans les flots de conception standard) ...

Depuis, les travaux sur la logique asynchrone ainsi que l'intérêt de la communauté scientifique ne cessent de s'intensifier, notamment via des conférences spécialisées telle que la conférence IEEE ASYNC (*International Symposium on Asynchronous Circuits and Systems*) qui publie et regroupe les dernières avancées dans le domaine depuis 1994.

2.2. Concepts de base de la logique asynchrone

Dans les circuits synchrones, le séquencement des opérations est assuré par la présence d'une horloge globale : toutes les opérations sont actionnées par un front de cette horloge. Ce style de conception s'appuie sur deux hypothèses fondamentales :

- Les signaux manipulés sont binaires. L'algèbre de Boole offre alors un cadre de conception maîtrisé qui simplifie leur implantation électrique.
- Le temps est discrétisé. Ceci permet de s'affranchir des fluctuations électriques transitoires ainsi que des problèmes de rétro-action et boucles combinatoires (les opérations s'effectuent entre deux fronts d'horloge).

A contrario, dans les circuits asynchrones, le codage binaire des signaux est conservé mais pas l'hypothèse de discrétisation temporelle. De ce fait, les circuits asynchrones constituent une classe plus large de circuits, les mécanismes de synchronisation pouvant être assurés par tout autre moyen alternatif à une horloge globale.

2.2.1. Principe de base

Le mode de fonctionnement asynchrone ne suppose pas qu'il y ait une relation temporelle *a priori* entre les événements. Les différents blocs sont activés par la présence de données à leurs entrées. La structure de base d'un circuit asynchrone est représentée dans Fig. 2.2.

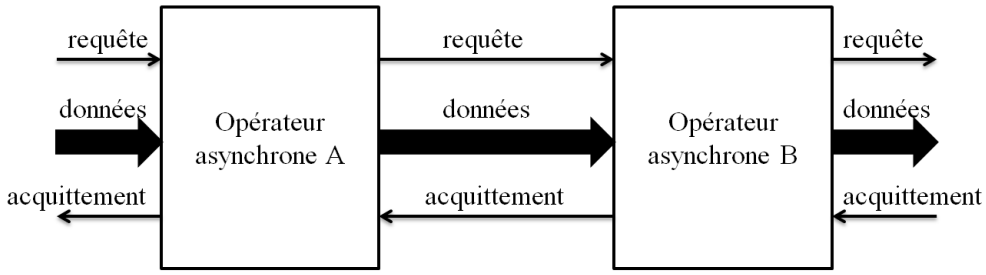


Figure 2.2 – Structure générale d'un circuit numérique asynchrone

Les éléments qui échangent des informations sont connectés entre eux par un canal de communication (au sens large), sans recourir à un signal externe de synchronisation. Le transfert d'informations est ainsi géré localement dans le circuit (ce type de fonctionnement est similaire à celui des systèmes de type "flot de données"). Chaque action de l'émetteur doit être acquittée par le récepteur avant que l'émetteur ne puisse agir à nouveau. Ce type de communication est dit "à poignée de main" (*handshake*) ou de type requête/acquittement.

Plusieurs protocoles existent pour implanter ce type de communication avec comme point commun le même objectif : détecter la présence de données sur le canal d'entrée, signaler la consommation de cette donnée, puis après son traitement, signaler la disponibilité du résultat sur le canal de sortie. Certains de ces protocoles, associés à un codage particulier des données, permettent d'implanter des systèmes insensibles aux temps de traitement.

2.2.2. Protocoles de communication et codage des données

Deux protocoles de communication sont couramment utilisés dans l'implantation de circuits asynchrones : le protocole 2 phases, dit NRZ (sans retour à zéro, *half-handshake*), et le protocole 4 phases, dit RZ (retour à zéro, *full-handshake*). Le protocole 2 phases correspond à la séquence minimale nécessaire à l'implantation d'une communication de type requête/acquittement. Son principe, décrit dans Fig. 2.3, s'appuie sur deux phases de fonctionnement :

- *Phase 1* : détection de la requête de l'émetteur par le récepteur, traitement de la donnée puis génération d'un signal d'acquittement. C'est la phase active du récepteur.
- *Phase 2* : détection du signal d'acquittement par l'émetteur (signalant la consommation d'une donnée). Le canal de communication est alors prêt à recevoir une nouvelle donnée. Il s'agit de la phase active de l'émetteur.

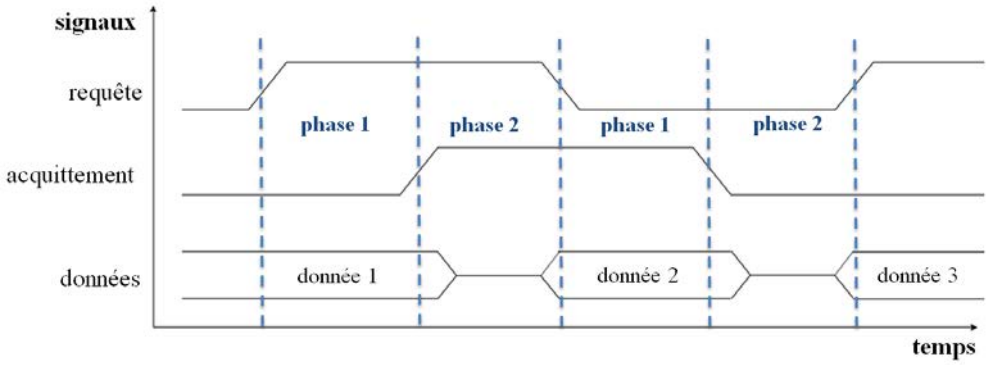


Figure 2.3 – Protocole de communication 2 phases

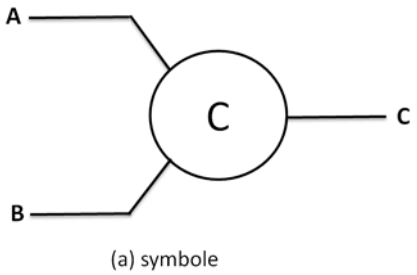
Tandis que dans la logique synchrone le codage des données est implicite, la logique asynchrone nécessite à la fois de coder les données mais aussi leur validité. Pour cela, deux solutions se distinguent :

- *Codage "données groupées"* : un signal de validité est explicitement ajouté en parallèle aux signaux de données. La valeur de la donnée est codée comme en logique synchrone, i.e. un rail par bit de donnée (*single rail*), et le signal de validité par un rail supplémentaire. L'inconvénient de ce type de codage est la sensibilité aux délais (l'occurrence de la génération du signal de requête est calibrée par rapport au chemin critique de l'opérateur).
- *Les codages insensibles aux délais* : l'information de validité est encapsulée avec le signal de donnée. Un bit de donnée requiert ainsi deux rails dont la combinaison contient à la fois la donnée et l'information de validité (*dual rail*). Deux codages sont communément adoptés : le codage trois états et le codage quatre états. Dans le codage 3 états, un rail est activé pour la valeur '1' et l'autre pour la valeur '0'. L'état "11" est interdit tandis que l'état "00" correspond à des données invalides. Ce codage est particulièrement adapté au protocole de communication 4 phases vu que le passage d'une valeur valide à une autre implique nécessairement le passage par l'état invalide. Dans le codage 4 états, la valeur de la donnée est codée dans la parité du nombre donné par les deux rails. Chaque fois qu'une donnée est

émise, la parité est changée, ce qui permet de passer d'une donnée à une autre sans passer par un état invalide. Ce type de codage est donc particulièrement adapté au protocole de communication 2 phases.

2.2.3. La porte de Muller

L'implantation des protocoles de communication asynchrones nécessite l'utilisation d'un opérateur réalisant la synchronisation de plusieurs signaux. Une telle fonction peut être réalisée avec des portes logiques de base mais moyennant un coût matériel relativement important. On utilise donc généralement une porte spécifique réalisant la fonction "rendez-vous" entre plusieurs signaux, appelée porte de Muller, ou *C-element*. Son symbole et sa table de vérité sont donnés dans Fig. 2.4. Lorsque les entrées d'une porte de Muller sont égales, leur valeur est transmise à la sortie. Lorsqu'elles sont différentes, la valeur précédente de la sortie est mémorisée. Cette porte implante donc naturellement la fonction "rendez-vous" entre plusieurs signaux.



A	B	C
0	0	0
0	1	Mem
1	0	Mem
1	1	1

(b) table de vérité

Figure 2.4 – Symbole (a) et table de vérité (b) d'une cellule de Muller

Il existe plusieurs implantations répondant à cette spécification générale, chacune ayant différentes caractéristiques de vitesse, consommation et surface. Dans [Iss10], un chapitre est dédié à la comparaison de ces différents types d'implantation en technologie CMOS 65nm. Fig. 2.5 représente le schéma transistor d'une des implantations les plus simples et les plus utilisées de la porte de Muller à deux entrées (nommée *weak-feedback Muller gate*). Les quatre transistors de gauche servent à l'établissement de la valeur de sortie quand les deux entrées sont égales, tandis que les deux inverseurs tête-bêches servent au maintien de cette valeur quand les entrées sont différentes.

2.3. Propriétés et avantages des circuits asynchrones

Cette section présente brièvement les caractéristiques spécifiques aux circuits asynchrones ainsi que les avantages liés à leur utilisation.

• **Absence d'horloge**

Il s'agit de la propriété la plus significative des circuits asynchrones et dont découle plusieurs avantages de leur utilisation. L'absence d'horloge dans les circuits asynchrones permet de s'affranchir de beaucoup de problèmes et contraintes liées à sa présence. Ceci est de plus en plus vrai étant

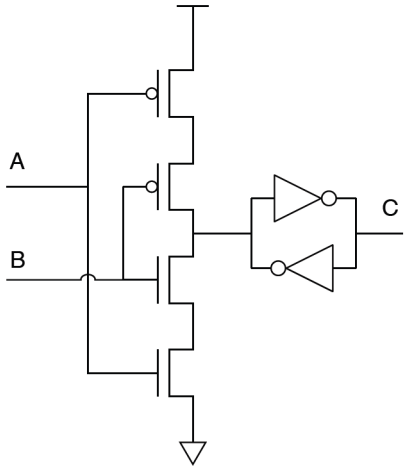


Figure 2.5 – Implantation *"weak feedback"* de la cellule de Muller

donné les développements récents des circuits numériques, avec en premier plan leur miniaturisation. Cette miniaturisation impose des contraintes de plus en plus importantes sur les circuits de distribution d'horloge qui deviennent prépondérants dans les systèmes synchrones en terme de complexité, surface et consommation. Leur conception peut vite devenir un facteur limitant les performances dans les circuits synchrones. *A contrario*, dans les circuits asynchrones, les éléments qui assurent le contrôle local des données sont distribués dans l'ensemble du circuit et sont donc plus faciles à maîtriser. Ceci élimine les pics de consommation liés au fronts d'horloge, la consommation globale du circuit est donc mieux répartie dans le temps ce qui limite considérablement le bruit dans les lignes d'alimentation.

De plus, pour certaines classes de circuits asynchrones, le fonctionnement est indépendant des délais introduits par les opérateurs, ce qui élimine d'emblée les problèmes liés au *clock skew* ainsi qu'à la gigue d'horloge (*jitter*). L'optimisation de la vitesse des circuits asynchrones repose donc uniquement sur l'optimisation des protocoles de communication et des cellules elle-mêmes.

• Calcul en temps minimal et robustesse aux conditions de fonctionnement

Les opérateurs asynchrones évaluent une fonction avec un temps variable qui dépend généralement des données en elles-même. De part les mécanismes de contrôle local cités précédemment, les données sont utilisables immédiatement après leur traitement tandis que, dans les circuits synchrones, le temps de traitement des données est limité par les temps les plus longs dans le circuit, ceux-ci peuvent en plus varier avec les conditions extérieures (température, tension d'alimentation) et la variabilité de procédé de fabrication. De ce fait, les circuits asynchrones sont potentiellement plus rapides et plus robustes que leurs homologues synchrones.

• Modularité et migration

L'utilisation de mécanismes de contrôle local dans les circuits asynchrones fait qu'ils sont naturellement très modulaires. Les différents blocs d'un système peuvent être assemblés comme des briques de base sans se préoccuper des problèmes de synchronisation. Ceci permet aussi une

distribution des tâches de manière plus aisée entre plusieurs concepteurs travaillant en parallèle sur un même système, ce qui favorise particulièrement la réutilisation de blocs et l'échange de propriétés intellectuelles (IP).

Le fonctionnement d'un circuit asynchrone est *a priori* indépendant de la réalisation des cellules qui le constituent pourvu que le protocole de communication soit respecté. Ainsi, il est facile de modifier l'implantation ou la technologie des cellules utilisées sans altérer l'aspect fonctionnel. Cette capacité de migration rend le style de conception asynchrone particulièrement attractif au vu de la rapidité des évolutions technologiques actuelles.

• **Faible consommation**

Cette propriété découle de plusieurs facteurs :

- *L'absence d'horloge* : dans les circuits récents, l'énergie dissipée par le système de distribution d'horloge et les éléments de mémorisation peut représenter jusqu'à 50% de la consommation d'un circuit.
- *La mise en veille* : un opérateur asynchrone qui ne traite pas de donnée ne consomme pas, les circuits asynchrones comprennent de manière intrinsèque une mise en veille locale des blocs qui ne fonctionnent pas à un moment donné.
- *L'absence d'aléas* : bien qu'ils ne sont pas gênants d'un point de vue fonctionnel dans les circuits synchrones, les aléas logiques impliquent quand même une consommation d'énergie supplémentaire inutile. La conception asynchrone suppose qu'il n'y ait aucun aléa afin d'obtenir des circuits corrects du point de vue fonctionnel.
- *Souplesse vis-à-vis des conditions de fonctionnement* : les circuits asynchrones étant plus robustes aux conditions de fonctionnement, il est aisé de réduire leur tension d'alimentation pour réduire leur consommation d'énergie.

2.4. Conclusion

Malgré le fort potentiel du style de conception asynchrone, son exploitation par les industriels est encore marginale par rapport au style de conception synchrone. Jusqu'au début des années 2000, il y avait un manque cruel d'outils de conception : ceux-ci ne couvraient pas l'ensemble du flot de conception et souffraient d'une mauvaise compatibilité avec les outils commerciaux. Ce n'est plus le cas actuellement : des outils et flots de conception qui s'intègrent avec les outils commerciaux existent maintenant. Le groupe CIS (*Concurrent Integrated Systems*) du laboratoire TIMA a développé l'outil TAST (*Tima Asynchronous Synthesis Tool*) dédié à la synthèse automatique de circuits QDI (*Quasi-Delay Insensitive*, il s'agit d'une classe de circuits asynchrones, *c.f.* l'annexe B) ou micropipelines. Depuis 2007, le développement de cet outil a été transféré dans une start-up grenobloise, la société Tiempo, qui propose un éventail d'IP asynchrones et un environnement de développement intégré dédié à la conception de circuits asynchrones (http ://www.tiempo-ic.com). Theseus Logic propose un flot de conception basé sur la logique NCL (*Null Convention Logic*), qui permet de spécifier entièrement des systèmes asynchrones. On peut également citer le flot de Handshake Solutions (Haste [1]). Du côté des universitaires, plusieurs outils ont été développés

1. http ://www.ispd.cc/slides/slides_backup/ispd06/8-2.pdf

ces dernières années ou sont en cours de développement. Par exemple, l'outil Petrify [2], développé à l'UPC (Universitat Politècnica de Catalunya), permet la synthèse de circuits asynchrones SI (*Speed Independent*) à l'aide d'un type particulier de réseau de Petri (*signal transition graphs*). On peut également citer les outils MINIMALIST [3] (développé par l'université de Columbia) et Balsa [4] (université de Manchester).

L'adoption de ces outils peut encore prendre un certain temps car les concepteurs spécialisés en logique asynchrone sont encore peu nombreux. L'autre facteur qui limite la démocratisation des technologies asynchrones est le manque de formation à ces techniques de conception alternatives. Le principe de discrétisation temporelle utilisé en conception synchrone est tellement ancré dans la manière de concevoir les systèmes qu'il est très difficile de le remettre en cause puisqu'on le retrouve à tous les stades de la conception. La conversion au mode de fonctionnement asynchrone demande une remise en cause profonde des habitudes des développeurs de circuits numériques, ce qui peut prendre quelques années.

3. Les oscillateurs en anneau asynchrones

Les anneaux auto-séquencés, issus des techniques de conception asynchrone, ont émergé au début du $20^{ième}$ siècle comme une sérieuse alternative aux anneaux à inverseurs classiques souvent utilisés pour la génération de signaux oscillants. Dans cette section, nous présentons d'abord l'architecture et fonctionnement d'un anneau à inverseurs classique. Ensuite, nous présentons l'architecture des oscillateurs en anneau auto-séquencés (ou anneaux de Muller), leurs principes et modes de fonctionnement.

3.1. Oscillateurs en anneau à inverseurs

Les oscillateurs en anneau à inverseurs sont des structures composées de chaîne d'inverseurs qui sont largement utilisées dans l'industrie pour générer des signaux d'horloge de part leur simplicité et bonne intégration dans les flots de conception standard. Bien qu'ils soient qualifiés d'asynchrones, ces oscillateurs n'utilisent pas de protocole de requête/acquittement comme ceux décrits dans la section précédente (les collisions de données sont possibles).

3.1.1. Architecture et principe de fonctionnement

L'architecture d'un anneau à inverseurs (*Inverter Ring Oscillator - IRO*) est représentée dans Fig. 2.6. L'anneau se compose d'une chaîne d'inverseurs rebouclée sur elle même. Chaque inverseur effectue la négation de la valeur du signal en entrée, ce qui peut donner lieu à la propagation d'événements (ou "fronts") au sein de l'anneau comme décrit dans Fig. 2.7. Chaque séquence (constituée des valeurs de sortie de deux étages successifs) de type "00" ou "11" correspond alors

2. http ://ceur-ws.org/Vol-827/16_OndrejGallo_article.pdf
3. http ://www.cs.columbia.edu/async/publications/minimalist-tech-report.pdf
4. http ://apt.cs.manchester.ac.uk/projects/tools/balsa/

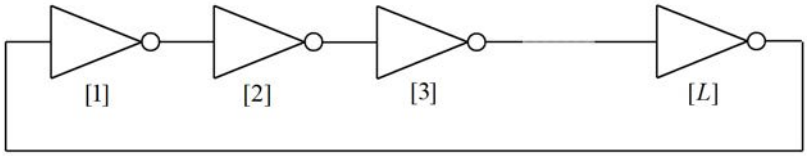


FIGURE 2.6 – Architecture d'un anneau à inverseurs

à un événement susceptible de se propager dans l'anneau. Chaque événement se propage après un délai, noté D, correspondant au temps de propagation d'un inverseur.

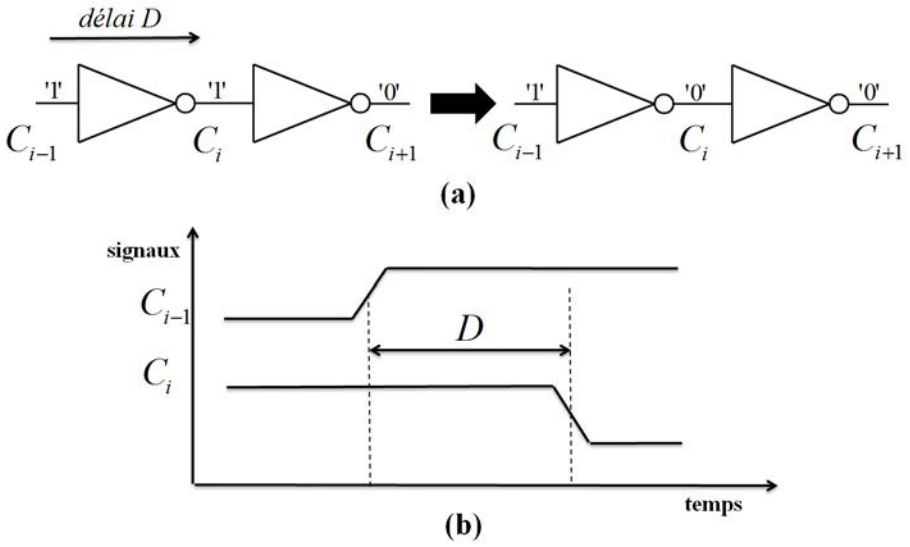


FIGURE 2.7 – (a) Propagation d'un événement présent à l'entrée de l'étage C_{i-1} vers l'entrée de l'étage C_i (b) Chronogramme de la propagation de cet événement

3.1.2. Modes de fonctionnement

Au niveau de leur comportement, en fonction de la parité du nombre d'étages, deux cas de figure peuvent se présenter :

- ***Le nombre d'étages est impair***

Le nombre d'événements à l'initialisation est alors nécessairement impair (par construction). Quelque soit ce nombre initial d'événements, le circuit finit par se stabiliser dans un mode d'oscillation stable. En effet, les différences de délais entre les différents inverseurs, aussi minimes soient elles (dues par exemple à la variabilité de procédé de fabrication), font que les événements finissent par se collisionner et s'annuler deux-à-deux. Par exemple, un anneau à 3 étages initialisé à l'état "000" (trois événements potentiels) finit nécessairement dans un état ne comportant qu'un événement (par exemple "010" ou "001"). Il apparait alors un comportement oscillatoire transitoire avec des

événements qui finissent par se rattraper dans le temps et s'annuler deux-à-deux. Ne subsiste alors qu'un événement qui se propage librement dans l'anneau, entrainant un comportement oscillatoire stable. Il s'agit ici de l'utilisation standard des anneaux à inverseurs pour générer un signal d'horloge. La fréquence d'oscillation d'un tel anneau est donnée par la formule suivante :

$$F_{osc} = \frac{1}{2LD}$$

où L est le nombre d'étages et D le délai de propagation d'un étage. La différence de phase entre deux étages successifs de l'anneau correspond au délai de propagation d'un étage D. Comme il n'y a qu'un événement qui se propage, alors la résolution de phase minimale de l'anneau correspond à ce délai D.

- ***Le nombre d'étages est pair***

Le nombre d'événements à l'initialisation est alors nécessairement pair. Comme dans le cas précédent, le circuit oscille dans un régime transitoire le temps que tous les événements se collisionnent et s'annulent deux-à-deux. Le circuit arrête alors d'osciller et se stabilise dans un état bloqué (par exemple un anneau à deux inverseurs finit toujours par se stabiliser dans l'état "01" ou "10"). Ce type de fonctionnement est exploité dans [FCN13] pour l'implantation de fonction physique inclonable (*Physical Unclonable Function - PUF*) et correspond au fonctionnement d'un circuit bistable. La durée des oscillations transitoires dépend principalement des asymétries dans les différentes branches de l'anneau et des pentes des signaux oscillants. Il existe également un cas théorique où ce type de circuit oscille indéfiniment. Ce comportement est observé en simulation si toutes les branches du circuit sont rigoureusement identiques (avec le même délai de propagation), les événements peuvent circuler indéfiniment sans se rattraper s'ils sont initialisés uniformément répartis dans la structure. Ce cas est néanmoins rarement observé en pratique : les légères asymétries dues à la variabilité de procédé de fabrication des transistors sont généralement suffisantes pour provoquer la collision d'événements et donc l'arrêt des oscillations.

3.1.3. Utilisation

Il est à retenir que ces anneaux sont la plupart du temps utilisés avec un nombre impair d'étages pour obtenir le comportement oscillatoire permanent attendu d'une horloge. Les étages sont souvent non initialisés puisqu'après un temps de démarrage et indépendamment de l'initialisation, il ne subsiste qu'un événement qui se propage durant le régime d'oscillation permanent. Il est à noter aussi qu'il existe plusieurs variantes des ces oscillateurs utilisant des portes plus complexes (pour permettre par exemple d'initialiser chaque étage indépendamment). L'anneau peut également consister en une seule porte inverseuse suivie d'un nombre quelconque d'éléments de retard.

L'implantation de ce type d'anneau est très simple et s'intègre parfaitement dans le flot de conception de circuits ASIC (*Application-Specific Integrated Circuit*). En ce qui concerne l'implantation dans les circuits reprogrammables, notamment les FPGA (*Field-Programmable Gate Array*), le concepteur dispose de plusieurs solutions pour réaliser des portes inverseuses et des éléments de retard. Ceux-ci peuvent être réalisés dans des blocs logiques programmables de type LUT (Look-Up-Table). Néanmoins, la précision en phase et fréquence de ce type d'implantation est fortement

conditionnée par la maîtrise des délais dans les lignes de retard qui sont souvent réalisées avec des structures complexes composées de multiplexeurs.

Les oscillateurs en anneau à inverseurs sont largement cités dans la littérature scientifique et ont été abondamment étudiés, modélisés et caractérisés. Ces oscillateurs sont *a priori* des circuits asynchrones, mais qui ne respectent pas les règles de conception décrits dans la section précédente. Le reste de cette section s'intéresse à un type d'oscillateur en anneau asynchrone qui respecte les règles de conception asynchrone, il utilise notamment un protocole de communication 2-phase pour gérer l'ordonnancement des évènements qui se propagent et pour éviter leur collision.

3.2. Oscillateurs en anneau auto-séquencés

Contrairement aux oscillateurs en anneau à inverseurs classiques, les oscillateurs en anneau auto-séquencés permettent la propagation simultanée de plusieurs événements sans qu'il y ait de collision grâce à des mécanismes de contrôle local dérivés des techniques de conception asynchrone. Cette section décrit ces anneaux et détaille leur comportement temporel.

3.2.1. Architecture et fonctionnement

L'oscillateur en anneau auto-séquencé est composé d'une chaîne de portes de Muller (telles que décrites dans la section 2.2.3) dont l'une des deux entrées est inversée. Il s'agit en réalité de la structure de contrôle d'un micropipeline tel que définie par I. E. Sutherland dans [Sut89], qui a été rebouclée pour former un anneau. Son architecture ainsi que la table de vérité d'un étage sont données dans Fig. 2.8. Chaque étage i est connecté à la sortie de l'étage précédent C_{i-1} via son entrée directe F_i (*forward input*), et à celle de l'étage suivant C_{i+1} via son entrée inversée R_i (*reverse input*).

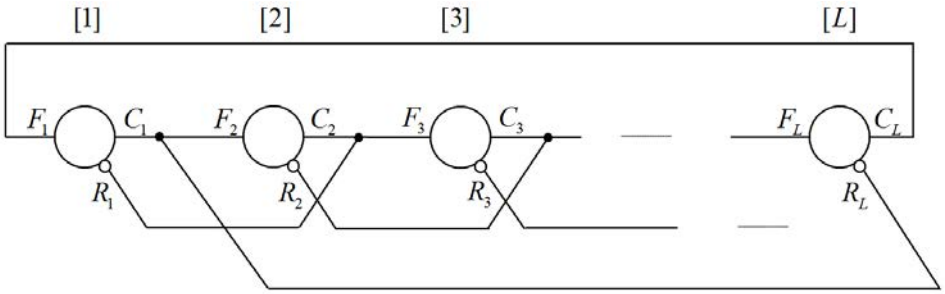


Figure 2.8 – Architecture d'un anneau auto-séquencé

Les étages du micropipeline communiquent entre eux via un protocole à poignée de main "2 phases" tel que décrit dans la section 2.2.2 de ce chapitre. Un événement (dans le sens transition électrique) sur une entrée F_i d'un étage correspond à une requête venant de l'étage précédent (C_{i-1}) tandis qu'un événement sur l'entrée R_i correspond à l'acquittement de l'étage suivant (C_{i+1}). Un événement sur la sortie de l'étage C_i sert à la fois de requête pour l'étage suivant (C_{i+1})

et d'acquittement pour l'étage précédent (C_{i-1}). Chaque requête et acquittement correspond au transfert d'une donnée (ici la donnée correspond à un événement) entre deux étages inter-connectés. Contrairement aux anneaux à inverseurs, plusieurs événements peuvent circuler dans l'anneau sans collision grâce au protocole à poignée de main.

L'anneau est initialisé avec N événements qui commencent à se propager durant un état transitoire. L'anneau finit par atteindre un régime permanent où les événements se réarrangent dans l'anneau dans l'une des deux manières suivantes : soient il forment un agrégat qui se propage d'une manière régulière dans l'anneau (mode d'oscillation en rafale - *Burst oscillation mode*), soient ils se dispersent de manière uniforme tout au long de l'anneau et se propagent avec un espacement temporel constant (mode d'oscillation régulier - *Evenly-spaced oscillation mode*). Fig. 2.9 représente les deux modes d'oscillations d'un anneau asynchrone. Ces deux modes d'oscillation sont stables et dépendent des paramètres statiques de l'anneau tels que le nombre d'événements choisi à l'initialisation et les délais de propagation d'un étage. Le mode d'oscillation régulier peut être utilisé pour générer des horloges configurables à haute précision fréquentielle dont la fréquence peut être réglée grâce au contrôle du nombre d'événements circulant dans l'anneau [EYRF10]. Une analyse de ces modes d'oscillation, des effets analogiques qui les provoquent et des modèles utilisés pour les décrire est proposée dans la section 4 de ce chapitre.

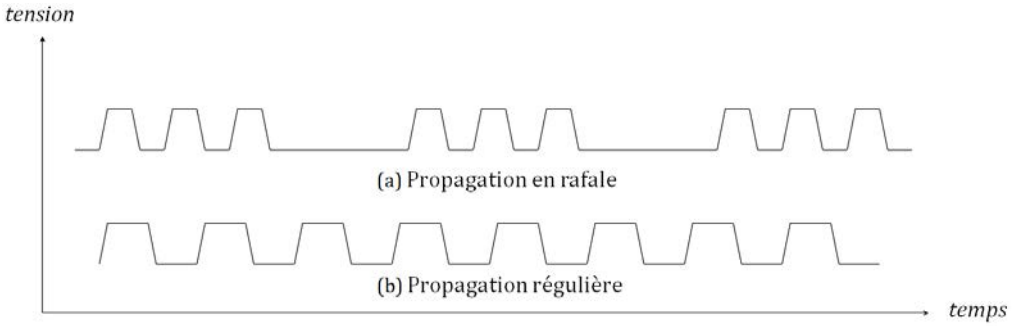


Figure 2.9 – Modes d'oscillation d'un anneau auto-séquencé

3.2.2. L'abstraction jetons/bulles

La suite de cette section présente la modélisation comportementale des anneaux asynchrones. Il est à noter qu'aucune hypothèse temporelle relative aux délais des portes logiques n'est faite à ce stade.

Le concept de bulles et jetons, dérivé du protocole de communication 2 phases tel que décrit dans [Ren10], offre un cadre qui facilite la représentation des données circulant dans un micropipeline asynchrone. Nous définissons les termes suivants :

– On dit qu'un étage i contient un jeton T (*token*) si sa sortie C_i est différente de celle de l'étage précédent C_{i-1}, un jeton représente alors la présence de donnée à l'entrée d'un étage :

$$\text{L'étage } i \text{ contient un jeton} \Leftrightarrow C_i \neq C_{i-1}$$

– On dit qu'un étage i contient une bulle B (*bubble*) si sa sortie C_i est égale à celle de l'étage précédent C_{i-1}, une bulle indique que l'étage est prêt à recevoir des données :

$$\text{L'étage } i \text{ contient une bulle} \Leftrightarrow C_i = C_{i-1}$$

Il est à noter que le nombre de bulles et jetons est fixé à l'initialisation de l'anneau et ne varie pas durant la propagation grâce aux mécanismes de contrôle local qui empêchent la collision de données (un jeton ne peut pas se propager à l'étage suivant si celui-ci contient également un jeton). Bien qu'elle soit généralement associée au protocole de communication 2-phases, l'abstraction jeton/bulles peut également être utilisée pour représenter les données circulant dans un anneau à inverseurs classique. Néanmoins, cela n'a pas d'intérêt concret étant donné que l'oscillateur en anneau à inverseurs ne conserve pas le nombre de données initial à cause des collisions.

3.2.3. Règles de propagation

A partir de la définition ci-dessus des bulles et jetons et de la table de vérité d'un étage de l'anneau (Fig. 2.8), on peut déduire la règle de propagation suivante :

– Un jeton présent dans l'étage i se propage à l'étage suivant $i+1$ si et seulement si l'étage $i+1$ contient une bulle (i.e. qu'il est libre et prêt à consommer une nouvelle donnée)

– De manière équivalente, une bulle présente dans l'étage $i+1$ se propage vers l'étage précédent i si et seulement si l'étage i contient un jeton (i.e. que lorsque l'étage $i+1$ consomme une donnée il libère l'étage précédent i, celui-ci est alors prêt à consommer une nouvelle donnée)

Fig. 2.10 représente la propagation d'un jeton dans un circuit de contrôle de micropipeline asynchrone entre un étage i et l'étage suivant $i+1$. La condition pour la propagation d'une donnée dans l'anneau peut donc être résumée par l'équation :

$$C_{i-1} \neq C_i \quad et \quad C_i = C_{i+1} \tag{2.1}$$

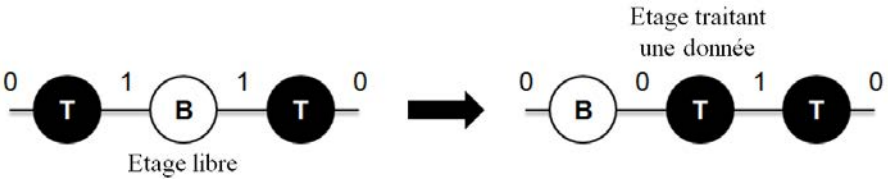


Figure 2.10 – Propagation d'un jeton dans un circuit de contrôle d'une micropipeline asynchrone

Ces règles impliquent que pour qu'il y ait propagation de données dans l'anneau, il faut que celui-ci contienne au moins une bulle et un jeton. Le nombre de jetons étant nécessairement pair (par construction), l'anneau doit donc contenir au moins 3 étages. Soient L le nombre d'étage de l'anneau, N_T le nombre de jetons et N_B le nombre de bulles, on a alors :

– $L \geq 3$, avec $L = N_B + N_T$

– $N_T \geq 2$, avec N_T pair

– $N_B \geq 1$

3.2.4. Modes de fonctionnement

Intuitivement, on peut s'attendre à ce que la forme du signal de sortie d'un anneau auto-séquencé dépende à la fois du nombre de jetons y circulant ainsi que de leur disposition initiale dans l'anneau. La réalité est tout autre, toutes les observations expérimentales et simulations électriques d'anneaux auto-séquencés montrent que les données se réarrangent naturellement dans l'anneau en fonction de leur nombre et de la taille de l'anneau, mais indépendamment de leur disposition initiale. Les observations montrent également qu'il existe deux modes de propagation : le mode «régulier» qui correspond à une propagation régulière des jetons (*evenly-spaced propagation mode*) et le mode «rafale» qui correspond à une propagation en rafale des jetons (*burst propagation mode*), tels que représentés sur Fig. 2.9. La durée du régime transitoire (pendant lequel les événement se "ré-arrangent") dépend principalement de la disposition initiale des jetons dans la structure. Par ailleurs le mode "régulier" est toujours obtenu pour une gamme de jetons continue (centrée ou non autour de la moitié du nombre d'étages en fonction des l'asymétrie de l'étage de base).

De manière plus générale, les circuits de type micropipeline constituent une classe à part entière de circuits asynchrones. Une part importante des recherches dans le domaine de la conception asynchrone s'est donc intéressé à l'explication et modélisation des modes de propagation des données dans leur circuit de contrôle afin d'optimiser les performances de ces systèmes. De ce fait, de nombreuses méthodes et approches ont été étudiées pour en évaluer et en optimiser les performances ([EFS98], [WG01], [GWG02],[HFMR08] et [Ham09]). La plupart de ces études ont pour objectif d'obtenir une propagation rapide et sans contention des données afin d'optimiser les débits. La section suivante de ce chapitre traite donc de la modélisation temporelle fine des anneaux auto-séquencés et des phénomènes analogiques rentrant en jeu.

4. Modélisation temporelle des anneaux auto-séquencés

Les simulateurs de circuits tels que SPICE utilisent des équations non-linéaires différentielles (*Ordinary Differential Equations - ODE*) pour prédire les comportements des circuits. Les modèles de type ODE prédisent assez bien les comportements des anneaux auto-séquencés observés dans l'expérience. Mais malgré leur précision, ces modèles sont trop détaillés et complexes pour comprendre les mécanismes de verrouillage dans l'anneau et comment contrôler l'état final de l'anneau (il faut une centaine de variables pour simuler un anneau à quelques étages).

Ebergen fut le premier à proposer l'utilisation de diagrammes *Charlie* pour prédire les performances de circuits de contrôle de micropipelines linéaires [EFS98]. Ce diagramme représente le délai de propagation d'une porte de Muller en fonction du temps qui sépare les événements à ses entrées. Il permet de prendre un compte un effet analogique crucial pour la compréhension de la manière dont les données s'y propagent (nommé effet *Charlie*). Winstanley poursuit l'étude en s'intéressant à ces circuits en boucle fermée (anneaux auto-séquencés) et en introduisant un second phénomène analogique appelé effet de *drafting* qu'il prend en compte dans le nouveau diagramme *Charlie* tri-dimensionnel [WG01]. Se basant sur ces deux effets analogiques, il explique pourquoi les anneaux auto-séquencés supportent les deux modes d'oscillation (rafale et régulier) et

définit un cadre mathématique qui lie l'occurrence de ces deux modes aux paramètres physiques de l'anneau (nombre d'étages, nombre d'événements initialisés, délais de propagation des portes et caractéristiques des effets *Charlie* et de *drafting*). A partir de là, plusieurs études ont été publiées, affinant ces modèles temporels ([HFMR08]) ou proposant des applications concrètes : génération de signaux à haute précision temporelle [FM04], anneaux asynchrones contraints évoluant toujours dans le mode de propagation régulière [HFMR09], oscillateurs programmables [YEZ+09], etc ...

Cette section décrit les effets analogiques propres à la cellule de Muller, la modélisation des anneaux auto-séquencés ainsi que de leur comportement temporel : verrouillage des modes "régulier" et "rafale", fréquence et distribution des phases.

4.1. Modèle temporel de la porte de Muller

Les portes de Muller sont des portes à deux entrées dont le délai de propagation dynamique est variable : il dépend à la fois du temps qui sépare l'arrivée des événements d'entrée mais aussi du temps écoulé depuis la dernière commutation de la sortie. Ces deux dépendances sont représentées par deux effets analogiques, appelés effet *Charlie* et effet de *drafting*.

4.1.1. L'effet *Charlie*

Le délai de propagation d'une porte de Muller est d'autant plus long que le temps qui sépare ses événements d'entrée est court. Ce phénomène a été nommé effet *Charlie* d'après Charles Edwin Molnar qui l'a en premier mis en évidence avec T. J. Chaney dans [CM73]. De manière générale, on peut attribuer ce phénomène à l'effet mémoire de la porte de Muller : lorsque ses entrées basculent avec un temps de séparation court, la porte doit basculer dans son mode "mémoire" un court instant (au moment ou la première entrée bascule) avant d'établir sa valeur logique finale (lorsque la deuxième entrée bascule).

Une observation fine des implantations électriques des portes de Muller permet de mettre en évidence ce comportement. Prenons par exemple l'implantation "weak-feedback" de la porte de Muller donnée dans Fig. 2.5. Supposons que la sortie C commute de '1' vers '0' suite à un événement sur son entrée A. Dans le cas où l'événement dans l'entrée A arrive longtemps après celui dans l'entrée B, les transistors PMOS et NMOS contrôlés par B ont déjà changé d'état et sont donc pour l'un complètement bloqué, et pour l'autre complètement passant. Le temps nécessaire à la transition de l'entrée A pour se propager au noeud interne entre les quatre transistors d'entrée et les deux inverseurs de sortie correspond donc uniquement au temps de blocage et de saturation des transistors contrôlés par A. Par contre, dans le cas où A arrive juste après B, le processus de blocage et de saturation des transistors contrôlés par B n'est pas achevé. Cela engendre alors un délai plus important pour que la transition sur A se propage à ce noeud interne, et finalement un délai de propagation global plus grand.

L'effet *Charlie* a une conséquence importante sur la propagation d'événements au sein d'un anneau auto-séquencé : il a tendance a faire en sorte que ceux-ci se repoussent mutuellement. En effet, un étage C_i libre entouré par deux étages C_{i-1} et C_{i+1} traitant des événements aura tendance à ralentir l'événement provenant de C_{i-1} puisqu'il aura un temps de propagation plus long (du

fait des deux événements proches provenant de C_{i-1} et C_{i+1}). Intuitivement, l'effet *Charlie* donne donc une première explication du mode de propagation régulier d'un anneau auto-séquencé :

Si un nombre important d'événements est contraint dans une petite structure, ceux-ci se repoussent mutuellement jusqu'à ce qu'il y ait effet de rétro-action et qu'un équilibre s'établisse (les événements se répartissent uniformément le long de la structure), provoquant le mode d'oscillation régulier d'un anneau auto-séquencé.

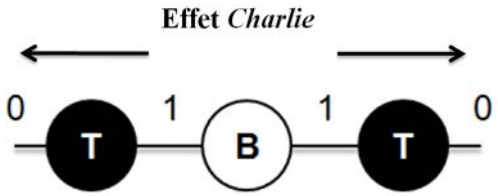


Figure 2.11 – Influence de l'effet *Charlie* sur la propagation de jetons dans un circuit de contrôle de micropipeline asynchrone

4.1.2. L'effet de *drafting*

L'effet de *drafting* est un phénomène plus général qui n'est pas spécifique à la porte de Muller et s'applique quasiment à toutes les portes logiques. Il représente l'impact du temps écoulé entre deux commutations successives de la sortie d'une porte sur son délai de propagation : plus ce temps est court, plus le délai de propagation est court. Ce phénomène est appelé effet de *drafting* par analogie au phénomène d'aspiration d'air dont tirent parti les coureurs automobile au moment de doubler une voiture. Cet effet est simplement dû à la capacité de sortie de la porte logique : si celle-ci commute trop rapidement, sa sortie n'a pas le temps d'atteindre complètement VDD ou GND avant une nouvelle commutation, ce qui engendre un temps de propagation plus court. Cet effet correspond donc à la prise en compte des pentes du signal analogique en sortie de chaque étage.

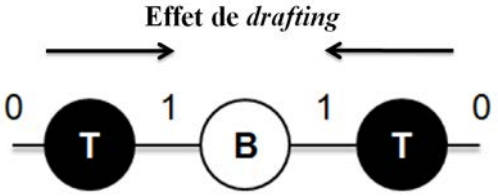


Figure 2.12 – Influence de l'effet de *drafting* sur la propagation de jetons dans un circuit de contrôle de micropipeline asynchrone

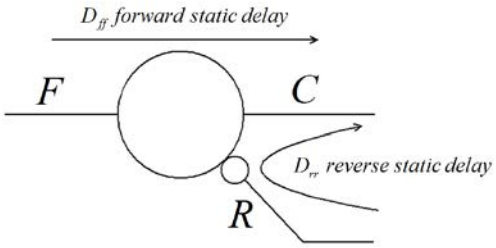
L'effet de *drafting* influe également sur la propagation d'événements dans l'anneau auto-séquencé : il fait en sorte que ceux-ci se rapprochent mutuellement. En effet, un étage de l'anneau

qui vient de voir passer un événement aura un temps de propagation plus court pour le prochain événement susceptible de le traverser. Ces deux événements se rapprochent donc dans le temps. De manière intuitive, on peut donc supposer que l'effet de *drafting* est la cause du mode de propagation en rafale d'un anneau auto-séquencé :

Un fort effet de drafting, associé à un faible nombre d'événements, peut provoquer la propagation en rafale dans un anneau auto-séquencé.

4.1.3. Le diagramme de *Charlie* 3D

Le modèle *Charlie* 3D, introduit par Winstanley dans [WG01], est un modèle temporel de la porte de Muller (ou de manière équivalent une porte de Muller avec entrée inversée) qui prend en compte les effets *Charlie* et de *drafting*. La structure et table de vérité de la porte étudiée sont représentés dans Fig. 2.13.



F	R	C
0	0	C^{-1}
0	1	0
1	0	1
1	1	C^{-1}

Figure 2.13 – Structure et table de vérité d'un étage d'anneau auto-séquencé

Fig. 2.14 représente un chronogramme d'un étage de l'anneau. Nous définissons dans un premier temps les variables associées à ce chronogramme des temps :

- s est le demi-temps de séparation entre les événements aux entrées F et R
$$s = \frac{t_F - t_R}{2}$$
- y est le temps entre la dernière commutation de la sortie C et l'instant moyen d'arrivée des événements aux entrées F et R
$$y = \frac{t_F + t_R}{2} - t_{C^{-1}} = t_{mean} - t_{C^{-1}}$$
- $charlie(s, y)$ est le temps de propagation de la porte vu à partir de l'instant moyen d'arrivée des événements aux entrées F et R
$$charlie(s, y) = t_C - \frac{t_F + t_R}{2} = t_C - t_{mean}$$
- D_{eff} est le temps de propagation effectif (ou apparent) de la porte qui correspond au délai de propagation de la porte dans le sens classique du terme
$$D_{eff} = charlie(s, y) - s$$

Ensuite, nous définissons un jeu de 5 paramètres qui régissent le modèle *Charlie* 3D :

- D_{ff} est le temps de propagation statique direct de la porte associé à l'entrée directe F (*Forward static delay*). Il correspond au délai de propagation de la porte après un événement sur l'entrée F associé à un temps de séparation suffisamment grand pour qu'il n'y ait pas d'influence de l'effet *Charlie*

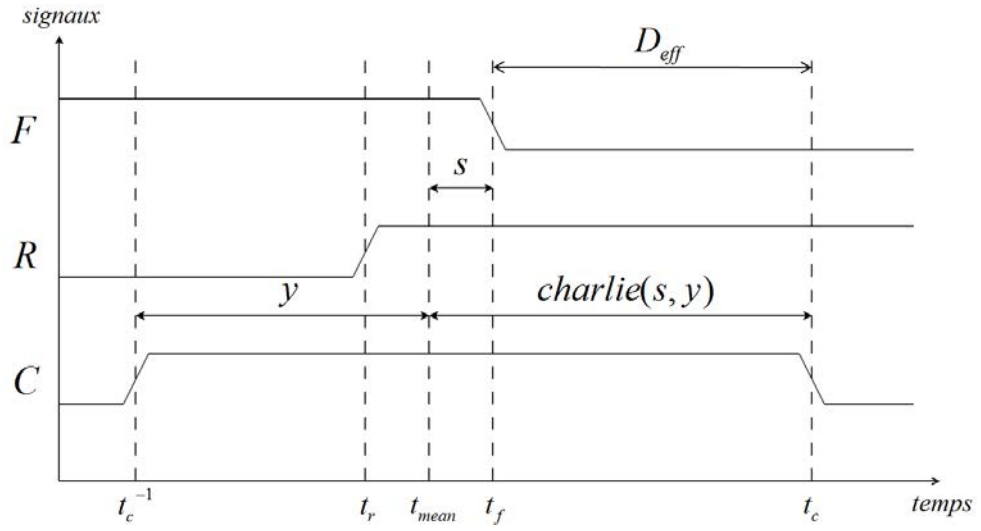


FIGURE 2.14 – Chronogramme de la cellule étudiée

- De manière similaire, D_{rr} est le temps de propagation statique inverse de la porte associé à l'entrée inversée R (*Reverse static delay*)
- $D_{charlie}$ correspond à l'amplitude de l'effet *Charlie* (en unité temporelle)
- τ est la durée de l'effet de *drafting* (en unité temporelle)
- A est l'amplitude de l'effet de *drafting* (en unité temporelle)

La forme analytique du modèle de *Charlie* 3D, telle que proposée par Winstanley dans [WG02] est exprimée par Eq. (2.2). Fig. 2.15 donne une représentation graphique tri-dimensionnelle de cette formule.

$$charlie(s,y) = D_{mean} + \sqrt{{D_{charlie}}^2 + (s - s_{min})^2} - A\exp^{-\frac{y}{\tau}} \tag{2.2}$$

D_{mean} et s_{min} sont liés aux délais de propagation statiques de la porte (s_{min} permet de prendre en compte la dissymétrie des portes logiques) :

$$D_{mean} = \frac{D_{rr}+D_{ff}}{2} \text{ et } s_{min} = \frac{D_{rr}-D_{ff}}{2}$$

La partie linéaire de Eq. (2.2) représente le délai de propagation "classique" de la porte, la partie en racine l'effet *Charlie* tandis que la partie exponentielle l'effet de *drafting*. La forme de la courbe *Charlie* 3D à y constant correspond à une parabole inscrite dans les droites d'équation $D_{ff} + s$ et $D_{rr} - s$. On remarque que le délai effectif de la porte (D_{eff}) est maximal lorsque le temps de séparation entre les événements en entrée est minimal ($s = s_{min}$ en prenant en compte la dissymétrie des portes), il s'agit de la zone où l'influence de l'effet *Charlie* est la plus importante. D'autre part, la forme du modèle de *Charlie* 3D à s constant suit la forme exponentielle de la charge d'une capacité à travers une résistance (*c.f.* Fig. 2.16).

Outre le fait qu'il prend en compte les effets *Charlie* et de *drafting* dans le temps de propagation d'une porte de Muller, ce modèle a d'autres apport significatifs. Premièrement, il élimine la discon-

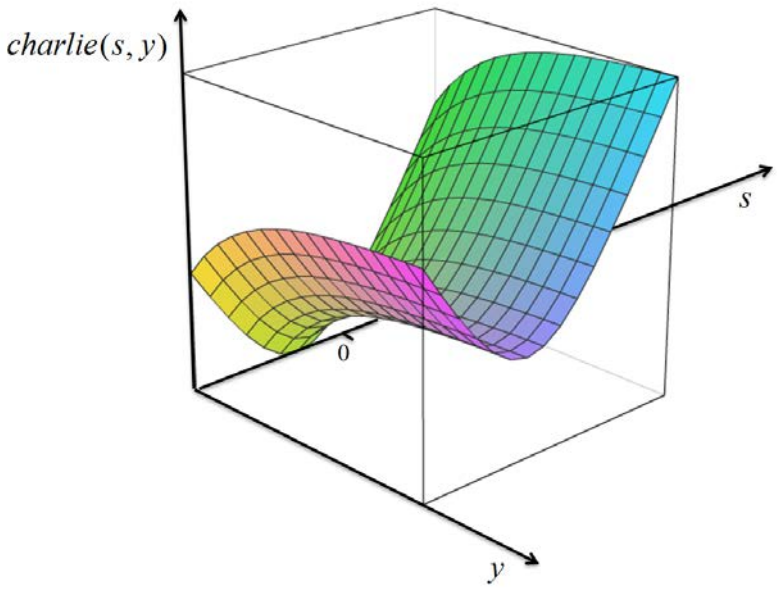


Figure 2.15 – Le diagramme *Charlie* 3D

tinuité du temps de propagation des modèles classiques de portes à deux entrées (lorsque $s = s_{min}$) qui n'est évidemment pas réaliste du point de vue physique. Deuxièmement, il permet de prendre en compte le temps de synchronisation des deux entrées (i.e. $|2s|$) en plus du temps de propagation effectif, ce qui est très utile pour l'étude de l'optimisation des anneaux auto-séquencés (pour notamment limiter les temps d'attente entre requêtes et acquittement et fluidifier la circulation de données dans un tel anneau).

4.2. Comportement en fréquence et distribution des phases

La fréquence d'un anneau auto-séquencé ne dépend pas directement de son nombre d'étages : elle dépend en premier lieu de son taux d'occupation (rapport entre le nombre d'événements et le nombre d'étages). De même, la résolution temporelle minimale et la distribution des phases d'un anneau auto-séquencé dépendent du nombre d'étages mais aussi du taux d'occupation de l'anneau. Cette section présente et analyse la courbe de fréquence d'un anneau auto-séquencé, la distribution des phases au sein de l'anneau ainsi que les mécanismes qui font que l'anneau oscille dans un mode d'oscillation régulier ou en rafale.

4.2.1. Calcul de la période d'oscillation

Dans [Ham09], Jérémie Hamon s'intéresse aux graphes d'états (ou diagramme de transition) d'anneaux auto-séquencés. Il identifie dans ces graphes les parcours correspondant à un mode d'oscillation régulier et ceux du mode d'oscillation en rafale et associe une annotation temporelle aux différents transitions (correspondant aux différents états de l'anneau). Ensuite, l'auteur s'intéresse

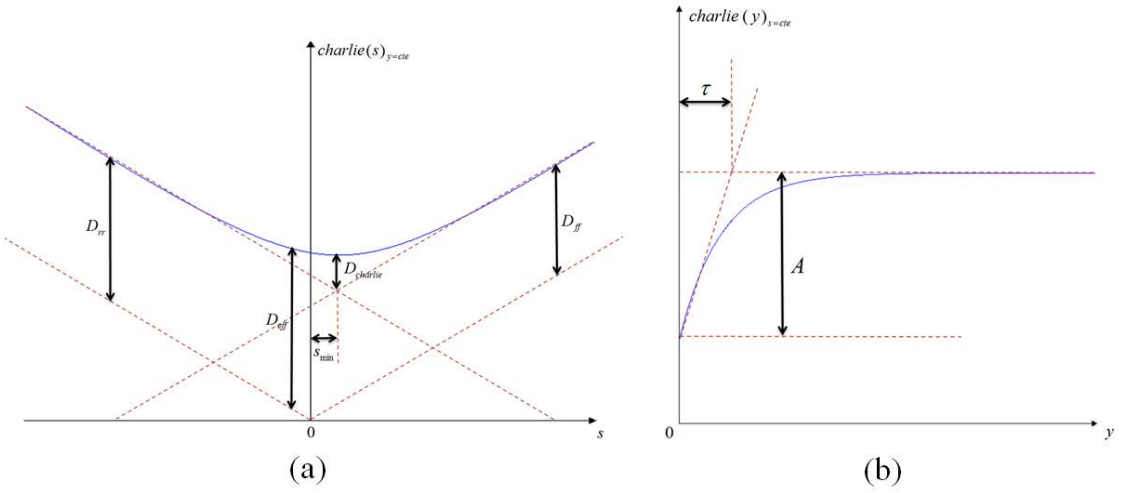


FIGURE 2.16 – (a) *charlie*(s) à y constant (b) *charlie*(y) à s constant

au cas particulier d'un anneau auto-séquencé à 5 étages et 2 jetons. Il fait l'hypothèse que l'anneau est en mode régulier de propagation et se sert de l'annotation temporelle du diagramme de transition de l'anneau pour calculer ses phases et sa période d'oscillation. Se basant sur la formule analytique obtenue et les résultats de simulation, il propose une relation empirique donnant la période d'oscillation T d'un anneau quelconque en fonction de son nombre de jetons N_T, nombre de bulles N_B, nombre d'étages L et paramètres de la courbe *Charlie* 3D :

$$T = 4 \times charlie(\frac{(N_B - N_T)T}{4L}, \frac{T}{4}) \tag{2.3}$$

4.2.2. Analyse de la courbe de fréquence

Comme expliqué précédemment, la fréquence d'un anneau auto-séquencé ne dépend pas directement de son nombre d'étages, elle peut être contrôlée par le nombre de bulles et de jetons à l'initialisation. En effet, la période d'oscillation d'un anneau auto-séquencé dans le mode régulier correspond au temps écoulé entre deux événements successifs traversant le même étage. Fig. 2.17 représente une courbe typique de la fréquence d'un anneau auto-séquencé en fonction de son taux d'occupation (nombre de jetons).

Les deux asymptotes de la courbes sont des droites d'équation déterminées par $F = \frac{1}{2D_{ff}(1+N_B/N_T)}$ et $F = \frac{1}{2D_{rr}(1+N_T/N_B)}$ et correspondent au comportement linéaire de l'anneau (i.e. lorsque l'influence de l'effet *Charlie* est faible). Par ailleurs, dans [Ham09] et [HFMR08], il est démontré que la fréquence maximale est obtenue lorsque le ratio entre nombre de bulles et nombre de jetons est égal au ratio entre le délais direct des portes et leur délai inverse :

$$\frac{N_T}{N_B} = \frac{D_{ff}}{D_{rr}} \tag{2.4}$$

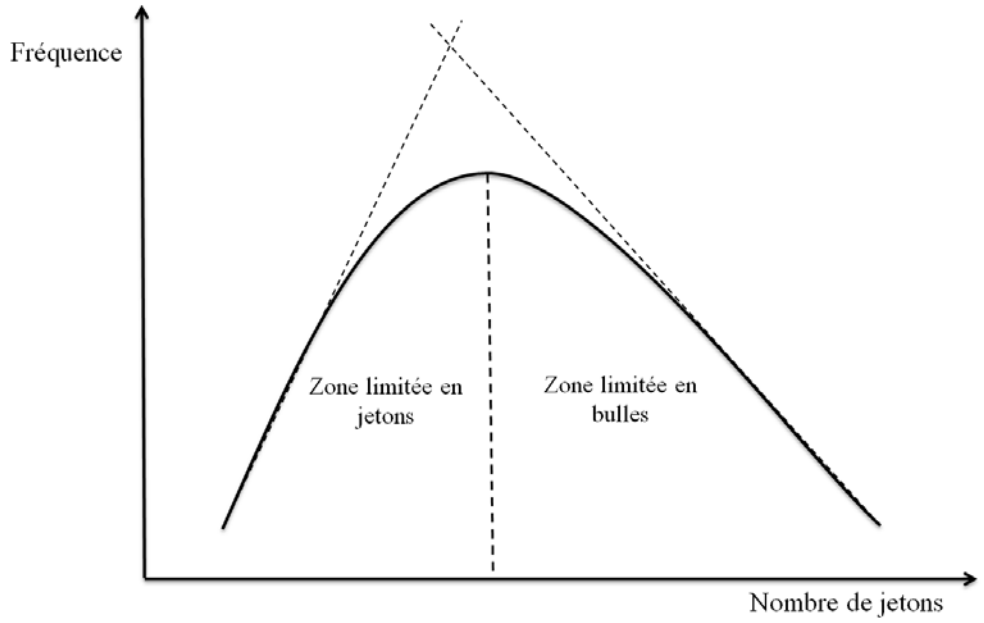


Figure 2.17 – Fréquence d'un anneau auto-séquencé dans le mode régulier en fonction de son nombre de jetons

Ce point partage la courbe en deux zones non nécessairement symétriques : une zone dite limitée en bulles où $\frac{N_T}{N_B} < \frac{D_{ff}}{D_{rr}}$, et une zone dite limitée en jetons où $\frac{N_T}{N_B} > \frac{D_{ff}}{D_{rr}}$.

Dans la zone limitée en jetons, le nombre d'événements effectif qui se propagent est égal aux nombres de jetons dans l'anneau : il y a suffisamment d'étages libres (bulles) pour que les événements se propagent sans contention. Dans ce cas, naturellement, la fréquence de l'anneau augmente en augmentant le nombre de jetons : il y a plus d'événements donc le temps écoulé entre deux événements successifs vu par un étage est plus court. Dans la zone limitée en bulles, c'est l'inverse qui se passe : il y a plus d'événements à traiter que d'étages libres (bulles), alors le nombre effectif d'événements qui se propagent correspond aux nombres de bulles. Dans cette zone, la fréquence diminue donc avec l'augmentation du nombre de jetons.

La zone avec une courbure non-linéaire (au milieu de Fig. 2.17) correspond à l'influence maximale de l'effet *Charlie*. La légère baisse de fréquence par rapport au modèle linéaire est due au fait que l'effet *Charlie* rallonge les délais de propagation dans sa zone d'influence maximale.

4.2.3. Distribution des phases

Les anneaux auto-séquencés sont naturellement adaptés pour la génération de signaux multi-phasés : cela est dû au fait qu'ils permettent la propagation de plusieurs événements au sein de la même structure et que ceux-ci - sous certaines conditions - se répartissent uniformément tout au long de la structure (mode d'oscillation régulier d'un anneau auto-séquencé). Dans un anneau à inverseurs classique, la résolution temporelle minimale qui peut être atteinte correspond au délai de

propagation d'un étage : il n'y a qu'un seul événement qui se propage avec un temps correspondant au délai de propagation d'un étage de l'anneau D, les transitions électriques qui occurrent dans les différents étages sont nécessairement séparées par un multiple du temps D. Tandis que dans un anneau auto-séquencé, plusieurs événements se propagent dans la structure : il est envisageable que le temps relatif entre deux événements soit inférieur au délai de propagation d'un étage.

Fig. 2.18 illustre la propagation régulière (*evenly-spaced propagation*) de deux événements dans un anneau auto-séquencé de 5 étages. Comme le montre ce chronogramme, la résolution de phase minimale de l'anneau $\triangle\varphi$ est une fraction du délai de propagation d'un étage (ceci n'est pas possible si un seul événement circulait dans l'anneau).

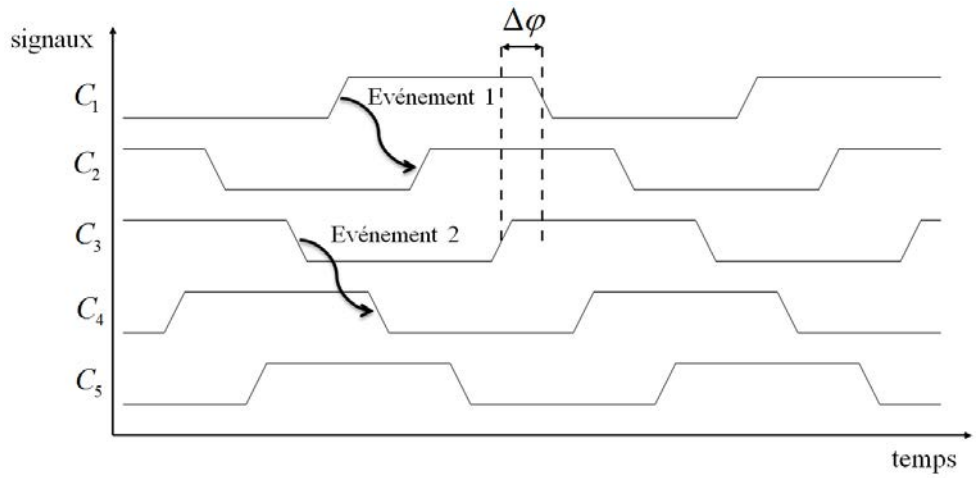


FIGURE 2.18 – Propagation de deux événements dans un anneau auto-séquencé de cinq étages

Considérons maintenant le cas général d'un anneau auto-séquencé de L étages avec N événements qui s'y propagent. Supposons que l'anneau atteint son régime permanent dans le mode d'oscillation régulier (*evenly-spaced oscillation mode*). Soient les paramètres suivants :

- T la période d'oscillation de l'anneau ($T/2$ est égal au temps écoulé entre deux événements successifs traversant le même étage)
- T_{tour} le temps que met un événement pour retraverser le même étage (c'est un multiple de T)
- D_{dyn} le délai de propagation direct dynamique dans le régime permanent

Le temps que met un événement pour faire tout le tour de la structure peut donc être exprimé ainsi :

$$T_{tour} = L \times D_{dyn} \tag{2.5}$$

D'autre part, durant ce temps T_{tour}, l'étage voit défiler N autres événements avant que l'un deux n'ai effectué une révolution complète. De plus le temps qui sépare deux événements successifs qui traversent le même étage est égal à $T/2$ (par définition de la période d'oscillation). Donc :

$$T_{tour} = N \times T/2 \tag{2.6}$$

D'après Eq. (2.5) et Eq. (2.6) on déduit la relation entre le délai dynamique en régime permanent, le nombre d'évenements de l'anneau et son nombre d'étages :

$$D_{dyn} = \frac{N}{L} \times T/2$$

Concrètement, D_{dyn} correspond à la phase (dans le domaine temporel) entre la sortie d'un étage et celle de l'étage suivant. La phase entre deux étages séparés de n étages est donc :

$$\varphi_n = n \times \frac{N}{L} \times T/2$$

Si on exprime les phases en degrés alors :

$$\varphi_n = n \times \frac{N}{L} \times 180^o \tag{2.7}$$

Selon cette équation, le nombre de phases différentes présentes dans l'anneau dépend directement du rapport N/L (ratio entre le nombre d'événements et le nombre d'étages). En effet, si L est un multiple de N, certains étages afficheront la même phase absolue. Le nombre d'étages affichant la même phase est d'autant plus important que le nombre de fois qu'il est possible de réduire le rapport N/L. Mais si jamais N et L sont premiers entre eux, alors l'anneau affiche autant de phases différentes et équidistantes que son nombre d'étages. Par exemple, un anneau à 9 étages et 4 événements affiche 9 phases équidistantes. Un anneau à 18 étages et 8 événements affiche également 9 phases équidistantes (car le rapport 8/18 est réductible à 4/9).

Corrollaire : Soit un anneau auto-séquencé de L étages avec N événements ayant atteint son régime permanent dans le mode d'oscillation régulier. Si L est un multiple de N alors certains étages peuvent afficher la même phase absolue. Si L et N sont premiers entre alors l'anneau présente autant de phases différentes et équidistantes que son nombre d'étages L. Dans ce cas la résolution de phase minimale de l'anneau (dans le domaine temporel) est exprimée par l'équation suivante :

$$\triangle\varphi = \frac{T}{2L} \tag{2.8}$$

Dans les anneaux auto-séquencés, la période d'oscillation ne dépend pas directement du nombre d'étages : il est possible de modifier leur fréquence en modifiant leur nombre d'événements sans changer le nombre d'étages. Si de plus ce nombre d'événements est choisi de façon à maintenir la configuration multi-phasée, alors on déduit d'Eq. (2.8) la propriété fondamentale suivante concernant la résolution de phase des anneaux auto-séquencés :

Propriété importante : *La résolution de phase d'un anneau auto-séquencé peut être réglée aussi fine que voulu en augmentant son nombre d'étages et en sélectionnant un nombre d'événements garantissant une fréquence haute tout en étant premier avec le nombre d'étages.*

4.3. Mécanismes de verrouillage et temps de démarrage

Nous proposons dans cette section d'analyser les mécanismes de verrouillage des modes régulier et en rafale d'un anneau auto-séquencé. Soit un anneau auto-séquencé initialisé avec N jetons. Pour simplifier la compréhension, nous supposons que l'anneau est limité en jetons (le raisonnement est

le même pour un anneau limité en bulles en remplaçant jeton par bulle), et que les étages de l'anneau sont symétriques ($D_{ff} = D_{rr}$). Il y a alors N événements susceptibles de se propager dans l'anneau (N jetons dans la zone limitée en jetons).

Supposons dans un premier temps que l'anneau ne contient que 2 jetons, et que ceux-ci sont initialisés "groupés" (dans des étages adjacents, i.e. "JJBBB...BB"). Fig. 2.19 illustre la propagation de ces deux jetons :

1) Le premier jeton commence à se propager dans l'anneau. Comme les étages suivants sont libres, il se propage avec un délai correspondant au délai statique direct des portes qu'il traverse (le temps de séparation vu par ce jeton est infini, l'effet *Charlie* n'agit donc pas).

2) Le second jeton suit le premier dès que celui-ci se déplace en laissant un étage de libre (bulle). Lors de son premier déplacement, le délai de ce deuxième jeton est le délai statique direct de la porte (le temps de séparation est grand car ce jeton a attendu que l'étage suivant se libère).

3) Par contre, lors de son second déplacement, les deux jetons se propagent quasi-simultanément : le retour d'acquittement (dû au déplacement du premier jeton) arrive quasiment en même temps que le signal de requête (dû à la présentation du deuxième jeton à l'entrée de l'étage, ceci implique un temps de séparation (s) très court au niveau de l'entrée d'un étage (et donc une forte influence de l'effet *Charlie*). Le deuxième jeton circule maintenant avec un délai $D_1 > D_{ff}$.

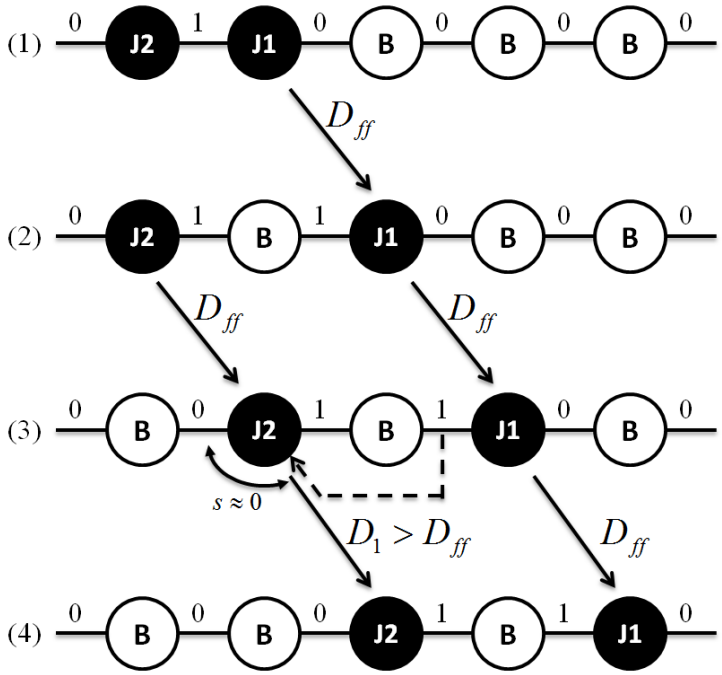


FIGURE 2.19 – Illustration des premières étapes de la propagation de deux jetons au sein d'un circuit de contrôle de micropipeline asynchrone

En pratique, le temps de propagation du deuxième jeton se stabilise rapidement autour d'une valeur D_{dyn} (correspondant à temps de séparation noté s_{hold}) : les deux jetons conservent entre

eux une distance de maintien (temps qui sépare le moment où le premier jeton traverse un étage et le deuxième traverse le même étage), notée D_{hold} qui dépend à la fois des effets *Charlie* et de *drafting* (mais aussi du rapport des délais statiques dans le cas général). Cette distance de maintien peut-être exprimée ainsi : $D_{hold} = D_{ff} + D_{dyn} + |s_{hold}|$, où D_{dyn} et s_{hold} correspondent au point de fonctionnement dans la courbe *Charlie* 3D où les effets *Charlie* et de *drafting* se compensent. $|s_{hold}|$ est le temps de "synchronisation" en boucle ouverte entre les deux événements.

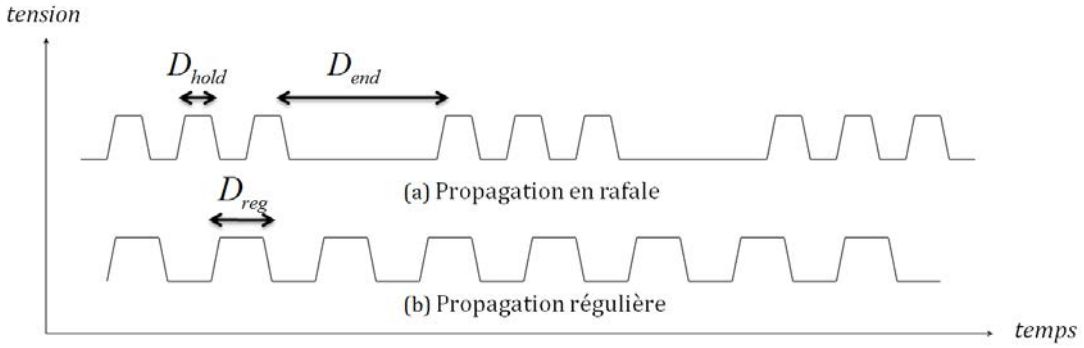


Figure 2.20 – Illustration des modes de propagation d'un anneau auto-séquencé

Maintenant, considérons le cas général où l'anneau est initialisé avec N jetons "groupés" (dans des étages adjacents, i.e. "JJJJ...BBBB"). Selon le raisonnement précédent, au bout de quelques cycles, les jetons se repoussent mutuellement en maintenant une distance D_{hold}. Notons alors D_{end} la distance temporelle entre le premier jeton et le dernier jeton (lorsque tous les jetons se sont repoussés en maintenant un distance D_{hold}). En fonction du taux d'occupation de l'anneau, il peut se présenter deux cas de figure (*c.f.* Fig. 2.20) :

- Si le taux d'occupation est faible, il est possible que le dernier jeton ne rattrape pas le premier ($D_{end} >> D_{hold}$) : le premier jeton ne subit pas de ralentissement et continue de se propager avec un délai égal à D_{ff} (par contre ceux qui le suivent sont ralentis et se propagent avec un délai $D_{dyn} > D_{ff}$). Dans ce cas, l'anneau se maintient dans ce régime permanent correspondant au mode de propagation en rafale comme montré dans Fig. 2.20. Le premier jeton n'a pas besoin d'attendre un signal de synchronisation, les jetons suivants ont un temps de synchronisation $|s_{hold}|$.
- Si le taux d'occupation est important, il est possible que le dernier jeton rattrape le premier ($D_{end} < D_{hold}$), dans ce cas le premier jeton subit une influence rétro-active des autres jetons et les jetons finissent par se répartir uniformément dans la structure. Moins il y a de jetons, plus les temps de synchronisation (en valeur absolue) sont grands, et plus la fréquence est basse. Le temps entre deux événements vus la sortie du même étage correspond alors à la demi période d'oscillation, il est exprimé ainsi : $D_{reg} = 2 \times D_{dyn} + |s_{dyn}|$. s_{dyn} et D_{dyn} correspondent à un point de fonctionnement sur la courbe *Charlie* 3D qui dépend du taux d'occupation de l'anneau (et absolument pas de l'état initial).
- La propagation des jetons est optimale lorsque les temps de synchronisation sont nuls. Dans ce cas d'étude ($D_{ff} = D_{rr}$), cela correspond à la situation où il y a autant de jetons que de

bulles dans la structure. Les étages contenant un jeton basculent tous simultanément et la fréquence est maximale.

Pour finir, nous proposons une simulation sous *Cadence* d'un anneau auto-séquencé à 7 étages avec 4 jetons initialisés dans la configuration "JJJJBBB". Comme le montre Fig. 2.21, les jetons se réorganisent pendant un régime transitoire qui dure quelques cycles d'oscillation avant de se propager avec un espacement constant dans le régime permanent. Il est à noter que le temps de démarrage d'un anneau auto-séquencé dépend de 3 facteurs : il est d'autant plus rapide que l'effet *Charlie* est fort, l'effet de *drafting* faible et la configuration initiale proche de la configuration finale [Ham09]. Dans le cas où l'anneau est initialisé avec une configuration proche de celle qu'il est sensé atteindre en régime permanent, alors la même simulation de l'anneau à 7 étages atteint son régime final en moins de 3 périodes d'oscillations. D'autre part, on peut remarquer que l'anneau présente autant de phases différentes que son nombre d'étages (7) étant donné que 4 et 7 sont premiers entre eux.

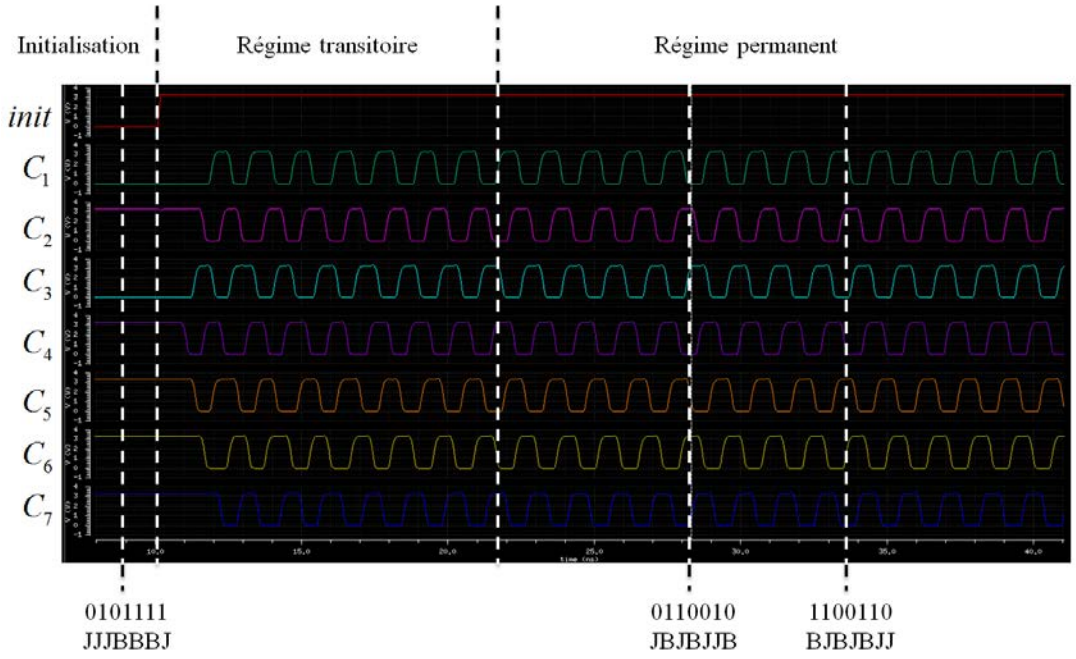


Figure 2.21 – Simulation sous *Cadence* d'un anneau auto-séquencé à 7 étages initialisé dans la configuration JJJBBBJ

5. Conclusion

A l'heure actuelle, les oscillateurs en anneaux à inverseurs restent très appréciés par les concepteurs de circuits car ils sont faciles à implanter et s'intègrent très bien dans les flots de conception standard. Leur précision en fréquence et les possibilité de reconfiguration dynamique numérique

qu'ils présentent restent néanmoins très limitées (une reconfiguration dynamique de leur fréquence nécessite de recourir à des composants analogiques). D'autre part, ils ne sont pas du tout adaptés à la génération de signaux multi-phasés : 1) leur résolution de phase minimale correspond au délai de propagation minimal d'un étage de la structure (cela est dû au fait qu'un seul évenement peut se propager dans la structure) 2) pour augmenter leur nombre de signaux multi-phasés, il faut nécessairement augmenter leur nombre d'étages ce qui cause une chute de la fréquence de ces signaux (ces structures permettent beaucoup de signaux multi-phasés basse-fréquence, ou bien peu de signaux multi-phasés haute-fréquence).

Ces dernières années ont vu l'émergence d'un nouveau type d'oscillateurs qui tire profit des techniques de conception asynchrone, notamment d'un protocole de communication de type requête/acquittement (protocole 2-phases tel que décrit dans la section 2.2.2), pour ordonnancer la propagation des données dans leur structure. Parmi les caractéristiques les plus intéressantes de ces oscillateurs, on peut citer :

- Contrôle automatique des *timings* entre événements se propageant au sein du même anneau sans avoir recours à des mécanisme de synchronisation externes (présence de mécanismes de vérouillage dûs aux effets *Charlie* et de *drafting*), d'où l'appellation "anneaux auto-séquencés"
- Réglage numérique dynamique de la fréquence d'oscillation : il est possible de modifier leur fréquence en injectant/retirant de manière dynamique des jetons/bulles dans la structure sans modifier leur structure (ceci se fait facilement en ajoutant des signaux d'initialisation aux étages de l'anneau)
- Adaptation à la génération de signaux multi-phasés : 1) La résolution de phase peut être réglée aussi fine que voulu (y compris une fraction du délai de propagation d'un étage) 2) La fréquence des signaux multi-phasés est indépendante du nombre de signaux qu'on veut générer

Comme le montre le chapitre 1, les oscillateurs numériques asynchrones sont omni-présents dans les générateurs de nombres aléatoires, en particulier ceux exploitant le *jitter* comme source d'aléa. Ces générateurs cherchent à échantillonner un signal *jitté* au plus proche de son moment de basculement pour détecter le *jitter* et générer un nombre aléatoire. Ceci requiert d'accumuler le *jitter* sur un intervalle de temps suffisamment long pour que son amplitude soit détectable [FMC85], ou bien de recourir à des mécanismes permettant de synchroniser les moments d'échantillonnage avec les fronts du signal *jitté*. Dans ce deuxième cas, ces mécanismes sont souvent soit très coûteux (PLL dans [FD02]), soit imprécis (se basant sur des modèles probabilistes) et qui influent sur l'extraction d'entropie en produisant du pseudo-aléa comme dans [SMS07]. A notre connaissance, [FD02] est la seule architecture qui permet un contrôle fin et précis des instants d'échantillonnage du signal *jitté*, mais elle requiert d'utiliser une PLL (relativement coûteuse) et ne permet pas des débits très importants (débits $\leq$ 2 Mbit/s).

A l'inverse, les anneaux auto-séquencés permettent non seulement de générer plusieurs signaux multi-phasés avec une très bonne précision fréquentielle, mais en plus ils permettent de contrôler les temps écoulés entre les fronts de ces signaux : la résolution temporelle avec laquelle on génère ces signaux peut-être réglée aussi fine que l'on veut. Ces propriétés semblent clairement exploitables pour proposer un nouveau principe de génération d'aléa permettant de contrôler l'entropie de

manière fine (en maîtrisant les instants d'échantillonnage des signaux *jittés*). Le TRNG à base d'anneau auto-séquencé (*Self-timed Ring Based TRNG - STRNG*), qui exploite ces propriétés est présenté dans le chapitre 3. Il s'agit de la contribution principale de ce travail de thèse.

Deuxième partie

Générateur de nombres véritablement aléatoires à base d'anneau auto-séquencé

Chapitre 3.

Présentation du générateur

Sommaire

1. Introduction

Le *jitter* est une source d'aléa dont il est possible de mesurer les caractéristiques, elle est donc très prisée pour la construction de générateurs testables avec la possibilité d'estimer l'entropie à leur sortie. Cependant l'une des principales difficultés pour son extraction est sa très faible amplitude, la résolution temporelle nécessaire pour sa détection doit être extrêmement fine (quelques picosecondes dans les technologies actuelles). Traditionnellement, trois principales approches sont adoptées dans la littérature pour contourner ce problème : l'accumulation du *jitter* (avec des débits lents et une mauvaise robustesse aux bruits déterministes), les techniques d'échantillonnage cohérent (débits relativement lents, nécessitent une PLL si on veut une bonne précision) et les techniques basées sur la somme de signaux *jittés* (rapides mais l'extraction d'entropie n'est pas suffisamment maîtrisée comme dans [Gol04] et [SMS07]).

Le principe de TRNG proposé dans ce chapitre permet de s'affranchir complètement du problème de l'amplitude du *jitter* en exploitant une des propriétés fondamentales des anneaux autoséquencés : leur résolution de phase peut être réglée aussi fine que voulu, simplement en augmentant le nombre d'étages de l'anneau L et en initialisant un nombre d'événements N premier avec L. Nous montrons qu'il est possible de tirer parti de cette propriété pour détecter le *jitter* de manière fine et générer des nombres aléatoires avec une entropie élevée.

Ce chapitre introduit le TRNG à base d'anneau auto-séquencé (*Self-timed ring based TRNG*), son principe de fonctionnement et son architecture. Il décrit ensuite la manière de garantir l'imprévisibilité de sa sortie en réglant les paramètres de l'extracteur d'entropie en fonction des caractéristiques mesurées de la source. Enfin, l'approche proposée est comparée avec celle utilisant des anneaux à inverseurs.

2. Principe du générateur

Le principe de base du STRNG reprend l'idée d'échantillonner un signal bruité (sujet au *jitter*) au plus près de son moment de basculement afin de générer un bit aléatoire. Comme il n'est pas possible de maîtriser précisément les moments d'échantillonnage dans les circuits, nous proposons d'échantillonner simultanément plusieurs signaux distribués dans le temps avec une résolution moyenne suffisamment fine par rapport à l'amplitude du *jitter*. Ce principe est illustré dans Fig. 3.1. Les signaux $(C_i)_{1\leq i\leq L}$ ont la exactement la même période d'oscillation T (ils sont synchronisés), et sont déphasés de $\triangle\varphi$ deux-à-deux. De plus, ils sont équi-répartis sur une demi-période d'oscillation ($L\times\triangle\varphi = \frac{T}{2}$). On suppose que les variations temporelles autour d'un front de chaque signal C_i suivent une loi gaussienne de moyenne t_i (telle que $t_i - t_{i-1} = \triangle\varphi$) et d'écart-type σ (le même pour tous les signaux).

Une horloge *clk* échantillonne simultanément tous les signaux C_i à un instant t. Par principe, quelque soit t, il existe un signal C_j tel que $|t-t_j| \leq \frac{\triangle\varphi}{2}$. Comme le montre la figure, si la résolution de phase $\triangle\varphi$ est suffisamment fine par rapport à l'amplitude du *jitter* (représentée par σ l'écart-type de la loi normale qui le modélise), alors ce signal C_j sera échantillonné suffisamment près de son moment de basculement pour générer un bit aléatoire. Pour un instant d'échantillonnage t fixé,

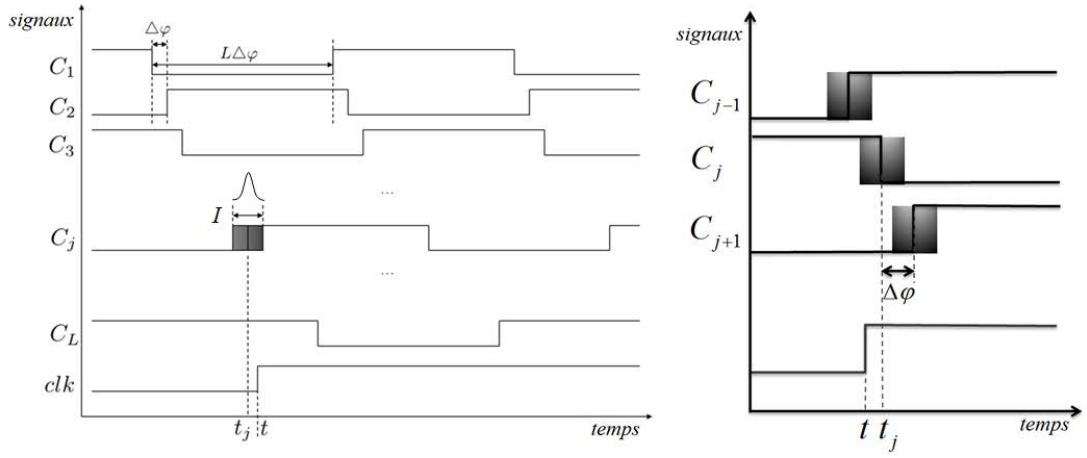


FIGURE 3.1 – Principe du TRNG à base d'anneau auto-séquencé (STRNG)

plus $\triangle\varphi$ est petit par rapport à σ , plus le bit généré est imprévisible.

Comme l'instant d'échantillonnage t n'est pas connu *a priori*, il n'est pas possible de déterminer quel signal C_j a été échantillonné au plus près de son front. L'idée consiste alors à sommer toutes les valeurs échantillonnées afin d'extraire le bit aléatoire parmi ceux qui ont une valeur déterministe. Dans le cas où l'instant d'échantillonnage est fixe (avec une fréquence synchronisée avec celles des signaux C_i) et connu, alors il n'est pas nécessaire de sommer tous les signaux (il suffit de récupérer la sortie de la bascule qui échantillonne ce signal).

3. Architecture du générateur

L'architecture qui réalise le principe ci-dessus est décrite dans Fig. 3.2. Elle se compose de deux parties : l'anneau auto-séquencé qui fournit la source d'entropie (*jitter*) et l'extracteur d'entropie qui génère en sortie les nombres aléatoires bruts. L'anneau auto-séquencé est le composant le plus important de cette architecture : il fournit les signaux bruités tout en permettant un contrôle fin de leurs instants moyens de basculement. Il est configuré dans un mode bien particulier. Premièrement, il fonctionne dans son mode régulier, le nombre de jetons est choisi de manière à fonctionner au plus près de la fréquence maximale :

$$\frac{N_T}{N_B} \simeq \frac{D_{ff}}{D_{rr}} \tag{3.1}$$

De plus, il est configuré dans son mode multiphase. Le nombre de jetons et le nombre d'étages sont choisis de manière à être premiers entre eux :

$$pgcd(N_T, L) = 1 \tag{3.2}$$

L'anneau fournit alors L signaux $(C_i)_{1\leq i\leq L}$ qui sont synchronisés entre-eux (avec une période T), équi-répartis sur une demi-période d'oscillation de l'anneau, et déphasés de $\triangle\varphi = \frac{T}{2L}$ comme le

montre la section 4.2.3 du chapitre 2. Les signaux de sortie de l'anneau doivent être ré-indexés pour correspondre à ceux de Fig. 3.1. C_i et C_{i-1} ne proviennent jamais d'étages adjacents, deux événements successifs dans le temps correspondent à la propagation de deux jetons dans des étages lointains en fonction du taux d'occupation de l'anneau. Par ailleurs, le temps moyen entre deux événements successifs $\triangle\varphi$ est déterminé par les effets *Charlie* et de *drafting* qui régulent la propagation des jetons dans un anneau auto-séquencé (*c.f.* la section 4.3 du chapitre 2). Enfin, le *jitter* autour de chaque événement correspond à celui généré par l'étage que traverse cet événement (et ne dépend pas des réalisations précédentes). Nous vérifierons cette hypothèse dans le chapitre 4 qui traite de la modélisation et de la caractérisation du *jitter* dans les anneaux auto-séquencés.

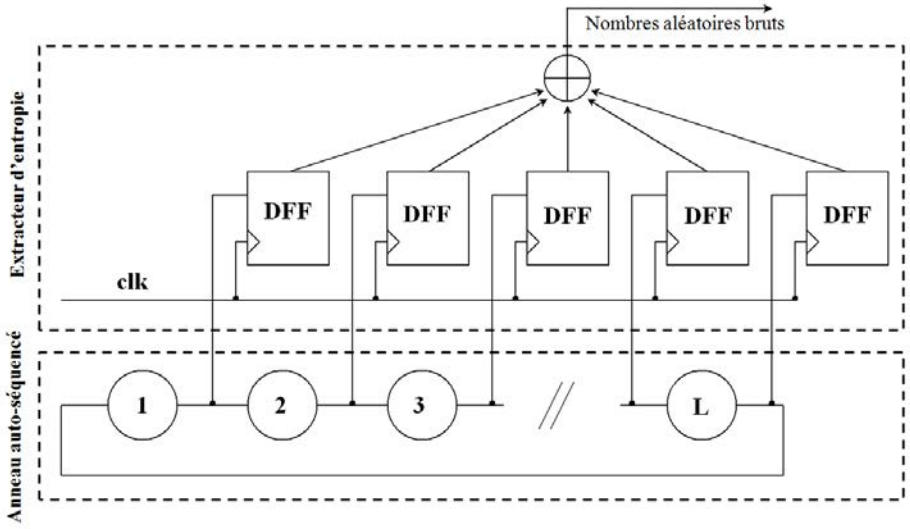


Figure 3.2 – Architecture du TRNG à base d'anneau auto-séquencé (STRNG)

L'extracteur d'entropie consiste en deux éléments : une série de L bascules et une fonction OU exclusif à L entrées. Les L bascules sont cadencées par la même horloge *clk* (celle-ci peut éventuellement provenir de l'anneau lui même). Leur signaux de sortie, notés $(C_i)_{1\leq i\leq L}$ sont sommés en base 2. Le OU exclusif est généralement implanté sous forme d'arbre constitué de rangées de portes OU exclusif standard avec des étages de mémorisation intercalés entre eux.

Enfin, pour corriger d'éventuels défauts statistiques, nous utilisons un filtre de parité comme celui présenté dans la section 4.4.1 du chapitre 1. Un filtre de parité d'ordre n réalise le OU exclusif de n bits d'entrée (en série) pour fournir un bit de sortie. Il permet de compresser les données et d'augmenter l'entropie par bit (si ceux-ci sont indépendants). Il est optionnel et dépend du dimensionnement du générateur (choix des paramètres de l'extraction en fonction des caractéristiques de la source et du niveau de sécurité désiré) mais il réduit son débit par n. Son architecture est donnée dans Fig. 3.3.

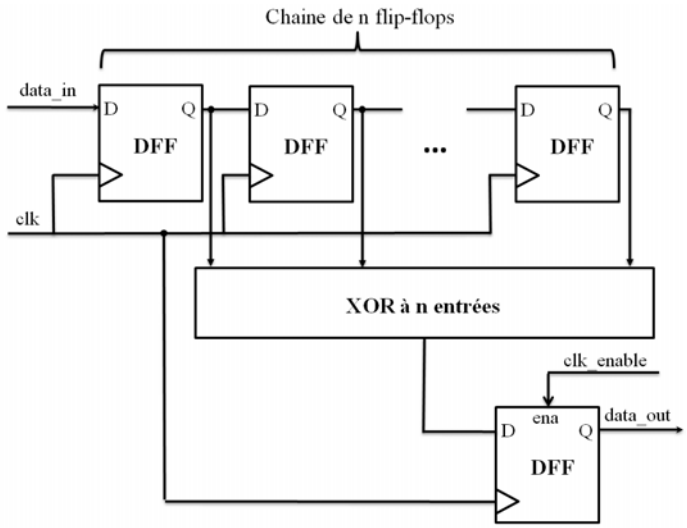


Figure 3.3 – Architecture du post-traitement arithmétique (filtre de parité d'ordre n)

4. Réglage et dimensionnement

On peut remarquer que l'utilisation de l'anneau auto-séquencé permet en théorie d'extraire l'aléa du *jitter* avec une entropie aussi grande que l'on veut quelque soit l'amplitude du *jitter*. Pour, cela il suffit d'affiner la résolution temporelle moyenne de l'anneau, ce qui peut être réalisé systématiquement en augmentant son nombre d'étages. Le dimensionnement de ce générateur consiste à régler finement le taux d'entropie en sortie en fonction de paramètres mesurés sur la source (chapitre 4) grâce à la modélisation du mécanisme d'extraction (chapitre 5) et cela en fonction du niveau de sécurité désiré.

Dans le chapitre 5, nous montrerons que l'entropie minimale par bit de sortie du générateur est directement liée à un ensemble de trois paramètres comme le montre Fig. 3.4 : le nombre d'étages L de l'anneau et sa période d'oscillation T (qui déterminent sa résolution temporelle) ainsi que l'écart-type σ du délai de propagation par étage dans le régime dynamique de l'anneau (qui est déterminé par le bruit dans le circuit et la configuration de l'anneau). L peut être directement contrôlé par le concepteur alors que T et σ peuvent être mesurés pendant la phase de développement du générateur. T ne dépend pas du nombre d'étages mais du rapport entre le nombre de jetons et le nombre d'étages (on peut maintenir une fréquence constante en augmentant le nombre d'étages). D'autre-part, nous montrerons dans le chapitre 4 que l'amplitude du *jitter* ne varie pas non plus avec le nombre d'étages.

Une stratégie de réglage et dimensionnement du STRNG peut donc être résumée en ces 4 étapes (d'autres stratégies sont proposées dans la section 2.8 du chapitre 5) :

1) Implanter un anneau, mesurer sa période d'oscillation et ses caractéristiques de *jitter*

2) Tracer la courbe d'entropie correspondante (Fig. 3.4) en utilisant le modèle que nous allons présenter dans le chapitre 5

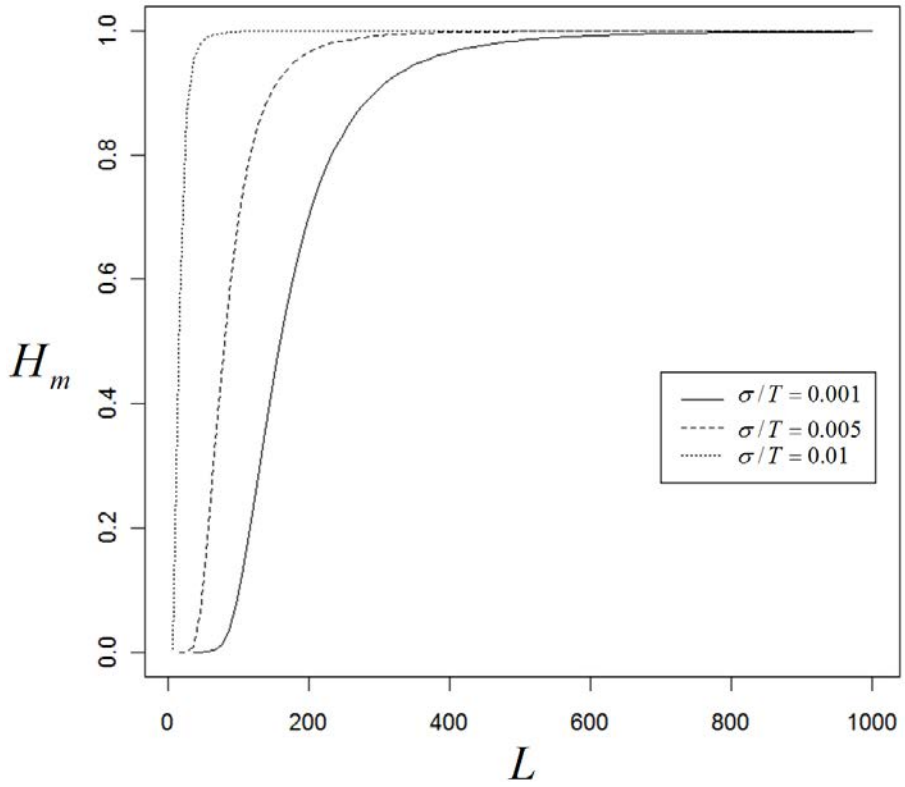


Figure 3.4 – Entropie minimale par bit de sortie du STRNG H_m en fonction de son nombre d'étage L, pour différents rapports σ/T

3) Choisir le nombre d'étages en fonction du taux d'entropie minimal désiré H_m (en fonction des exigences de l'application et des contraintes de taille)

4) Choisir le taux de compression du post-traitement arithmétique (ordre du filtre n) en fonction de H_m afin d'éliminer le biais statistique si ($H_m < 0.99$)

L'application de cette stratégie permet de s'assurer de l'imprévisibilité de la sortie du générateur quelque soient les paramètres de la source. En effet, ceux-ci peuvent beaucoup varier en fonction de la technologie et des implantations, cela doit être nécessairement pris en compte par les concepteurs, et cela de manière générale pour tous les TRNG (ce n'est malheureusement pas toujours le cas aujourd'hui).

5. Modes de fonctionnement

Le TRNG à base d'anneau auto-séquencé (STRNG) permet deux modes de fonctionnement en fonction du choix de l'horloge d'échantillonnage.

Mode externe - Dans ce mode de fonctionnement, l'horloge d'échantillonnage est indépendante de l'anneau auto-séquencé. Elle provient bien sûr de l'intérieur du circuit (pour des raisons de

sécurité) mais elle n'est pas synchronisée avec l'anneau auto-séquencé. Les instants d'échantillonnage relatifs aux fronts de l'anneau varient et les étages qui génèrent l'aléa ne sont pas les mêmes entre deux échantillonnages de l'anneau. L'entropie par bit varie en fonction des instants d'échantillonnage mais elle est toujours supérieure à la limite basse définie en fonction de la résolution de phase de l'anneau et de l'amplitude du *jitter*.

Mode interne Dans ce mode de fonctionnement, l'horloge d'échantillonnage provient de l'anneau lui-même, elle est donc synchronisée avec les signaux qu'elle échantillonne. L'instant d'échantillonnage relatif aux événements de l'anneau n'est pas connu, mais il est fixe. L'entropie par bit est donc constante (et aussi supérieure à la limite basse). Seuls quelques étages génèrent de l'aléa et ce sont toujours les mêmes.

En théorie, si le taux d'entropie minimal par bit est réglé suffisamment haut, alors ces deux modes sont complètement équivalents. En pratique par contre, les différences apparaissent pour un STRNG sous-dimensionné (*i.e* avec une entropie pas suffisamment élevée). Si la résolution de phase est très mauvaise, dans le mode externe, le signal de sortie fera apparaître un motif correspondant à la déviation de deux signaux d'horloges indépendants (celui de l'anneau et celui de l'horloge qui échantillonne). Par contre, dans le mode interne, le signal affichera toujours la même valeur fixe.

Le mode interne est donc extrêmement intéressant pour le *monitoring* de l'entropie du générateur, il permet de vérifier le réglage de phase en lien avec l'amplitude de *jitter*. D'autre part, il peut être appliqué pendant la phase de développement pour vérifier la répartition des phases sur la période d'oscillation de l'anneau en vérifiant successivement les signaux de sortie de l'anneau. Le mode externe, quant à lui, correspond au mode de fonctionnement classique de l'anneau dans une situation plus générale où le TRNG est intégré dans un système avec une horloge globale.

6. Comparaison avec l'approche utilisant des anneaux à inverseurs

Nous insistons sur le fait que le principe décrit dans la section 2 ne peut absolument pas être réalisé avec des anneaux à inverseurs. Si on utilise les sorties d'un anneau à inverseurs pour générer les signaux $(C_i)_{1\leq i\leq L}$, leur résolution de phase sera toujours limitée par le délai de propagation d'un étage (par principe dans les anneaux à inverseurs). Ce dernier est beaucoup trop grand par rapport à l'écart-type du *jitter* (le rapport est généralement proche de 1000). D'autre part, si on utilise plusieurs anneaux à inverseurs pour générer ces signaux, ceux-ci ne seront pas synchronisés (principe du RO-TRNG). Bien que l'architecture du STRNG semble *a priori* assez similaire à celle du RO-TRNG, le principe n'est pas rigoureusement le même et le STRNG apporte des améliorations sur plusieurs aspects :

Transparence de l'extraction d'aléa Dans le RO-TRNG, l'extraction d'entropie génère des séquences pseudo-aléatoires qui peuvent passer les tests statistiques sans réelle source d'entropie (*c.f.* la section 4.8.3 du chapitre 1). Cette séquence pseudo-aléatoire correspond à la déviation d'un nombre très important de signaux désynchronisés qui apparait sur le signal (théorique) correspondant à leur somme en base 2. Le comportement statistique de ce générateur ne reflète donc

absolument pas son taux d'entropie réel. Dans le cas du STRNG, les signaux sommés sont synchronisés. Le seul pseudo-aléa qui apparaît dans le signal de sortie correspond à la dérive de phase de l'horloge d'échantillonnage par rapport au signal de l'anneau (uniquement dans le mode externe). Il est important de préciser que le signal issu de la dérive de deux horloges génère des nombres qui ne passent aucune batterie de tests (même les moins exigeantes). Par ailleurs dans le mode interne, aucun pseudo-aléa n'est généré. Sans source réelle d'entropie (le *jitter*), le signal de sortie affiche toujours la même valeur. Dans ce cas, le comportement statistique reflète bien l'entropie réelle du générateur (ceci est montrée dans le chapitre 6).

Maîtrise de l'entropie L'entropie ne peut être maîtrisée de manière fine dans le RO-TRNG (*c.f.* la section 4.8.3 du chapitre 1) et son estimation repose sur des hypothèses difficilement vérifiables. Nous montrerons dans le chapitre 5 que l'entropie minimale par bit du STRNG peut être réglée très précisément en modifiant simplement le nombre d'étages de l'anneau en fonction de l'amplitude du *jitter* mesurée dans le circuit.

Sécurité Le RO-TRNG présente des failles de sécurité importantes principalement liées aux phénomènes de *locking*, ce qui les rend très vulnérables aux attaques (notamment électromagnétiques). Dans le chapitre 5, nous présenterons et analyserons un modèle de menace spécifique au STRNG et nous proposerons une série de contre-mesures adaptées.

Implantation Le RO-TRNG est plus facile à implanter que le STRNG. Nous verrons dans le chapitre 4 que l'implantation d'anneaux auto-séquencés est plus délicate que l'implantation d'anneaux à inverseurs. Globalement, le RO-TRNG peut souvent être implanté uniquement à l'aide d'un code VHDL (sans se préoccuper du placement et routage des éléments). Le STRNG nécessite un effort supplémentaire pour le placement et routage des cellules. Cependant, cette étape s'applique à une cible/technologie donnée et non à chaque implantation (ce n'est donc pas contraignant du point de vue industriel).

Taille Il est difficile de comparer des *designs* à taux d'entropie différents. Cependant, le modèle présenté dans [SMS07] fait l'hypothèse qu'il faut (statistiquement) $Nlog(N)$ oscillateurs pour remplir N intervalles égaux avec au moins un événement. Dans le cas du STRNG, il faut un seul anneau de N étages pour réaliser cela de manière sûre (non probabiliste). Ceci est important car cela veut dire que quand le *jitter* devient de plus en plus petit en amplitude, le nombre d'oscillateurs requis augmente rapidement dans le RO-TRNG, alors que dans le cas du STRNG c'est uniquement le nombre d'étages qui évolue de manière linéaire.

Consommation La consommation est globalement réduite dans le cas du STRNG principalement grâce au fait que l'arbre de XOR présente beaucoup moins d'activité que dans le RO-TRNG. Dans ce dernier, la déviation de phase entre les multiples oscillateurs provoque une activité continue dans l'arbre de XOR (toutes les branches basculent à intervalles réguliers). Dans le STRNG, notamment dans le mode d'utilisation interne, seulement quelques bascules changent de valeur, l'activité de l'arbre du XOR est donc fortement réduite.

7. Conclusion

Dans le TRNG proposé dans ce chapitre, la source d'aléa est le *jitter* d'événements se propageant au sein d'un anneau auto-séquencé. L'extraction d'entropie se fait en échantillonnant ces événements à l'aide de bascules, elle est séparée de la source d'un point de vue architectural. Ainsi, la source d'entropie peut être mesurée et caractérisée à l'endroit même où s'effectue l'extraction d'entropie (sorties de l'anneau), ce qui permet de construire un modèle du STRNG et d'estimer l'entropie de ses bits de sortie. Celle-ci est réglée directement en modifiant le nombre d'étages de l'anneau.

L'utilisation de l'anneau auto-séquencé permet de résoudre la problématique de la faible amplitude du *jitter*, elle apporte deux avantages significatifs par rapport aux techniques classiques utilisant des anneaux à inverseurs : la possibilité d'extraire l'aléa à partir du *jitter* systématiquement quelque son amplitude (en maitrisant le taux d'entropie en sortie) ainsi que des possibilités de *monitoring* avancées grâce au mode interne (qui devrait permettre une implantation simple des tests et alarmes internes).

Dans les trois chapitres suivants, nous allons caractériser le *jitter* dans les anneaux auto-séquencés. Se basant sur cette caractérisation, nous présenterons un modèle pour estimer l'entropie du STRNG. Ensuite nous proposerons une série d'extensions de sécurité pour le générateur (tests et alarmes) et enfin nous évaluerons son coeur dans des cibles ASIC et FPGA.

Chapitre 4.

Modélisation, implantation et caractérisation d'anneaux auto-séquencés

Sommaire

1. Introduction

La construction d'un modèle pour le STRNG nécessite de connaître les caractéristiques de la source d'aléa (propriétés statistiques du *jitter*) et une analyse approfondie de la manière dont les variations temporelles peuvent se propager dans la structure (car cela peut conditionner d'éventuelles dépendances entre les bits du sorties du générateur). D'autre part, le dimensionnement du STRNG nécessite de mesurer la période des anneaux qu'on souhaite implanter ainsi que l'écart-type du temps de propagation de leurs étages dans le régime permanent. Or, il n'existe pas de moyen direct pour mesurer celui-ci : nous avons accès uniquement à l'écart-type sur la période d'oscillation (celui-ci peut-être mesuré directement sur un signal de sortie de l'anneau). Un des objectifs de ce chapitre est donc d'établir le lien entre l'écart-type du temps de propagation des étages dans le régime permanent (*jitter* par étage) et celui de la période d'oscillation de l'anneau (*jitter* sur la période d'oscillation).

Alors que beaucoup de travaux se sont intéressés aux propriétés du *jitter* dans toutes sortes d'oscillateurs, il n'existe pas encore de travaux concrets traitant des anneaux auto-séquencés. On peut se poser diverses questions quant à la nature du *jitter* dans ces anneaux, et l'influence que peuvent avoir dessus les effets *Charlie* et de *drafting*. Dans le contexte de ce travail de thèse, nous nous intéressons à deux aspects principaux : premièrement, observer le comportement d'anneaux auto-séquencés dans différentes technologies, deuxièmement, qualifier la nature du *jitter* dans ces anneaux et le quantifier. La section 2 introduit et définit le *jitter*. Le reste du chapitre se compose de trois parties indépendantes :

1) *Section 3* : Cette section propose une modélisation haut-niveau (en VHDL) des anneaux auto-séquencés prenant en compte les phénomènes temporels des portes (effets *Charlie* et de *drafting*) ainsi qu'une modélisation du *jitter*. A l'aide de ce modèle, nous analysons la manière dont des variations temporelles se propagent au sein de l'anneau et nous mesurons l'écart-type de la période d'oscillation en fonction de différents paramètres de l'anneau.

2) *Section 4* : Cette section traite de l'implantation d'anneaux auto-séquencés dans deux familles de FPGA (Altera Cyclone III et Xilinx Virtex 5), et de leur caractérisation en terme de fréquence, *jitter* et résistance aux variations de tension.

3) *Section 5* : Dans cette section nous présentons le premier circuit implanté ASIC dans le cadre de ce travail (le second contient des coeurs du STRNG, il est présenté dans le chapitre 6). Il a été réalisé dans une technologie AMS (*Austrian MicroSystems*) CMOS 350nm et contient huit configurations d'anneaux auto-séquencés à des fins de caractérisation.

Enfin, la section 6 conclut ce chapitre.

2. Notions de base sur le *jitter* et sa quantification

Le *jitter* peut être défini comme la déviation temporelle des fronts d'un signal numérique par rapport à leurs instants idéaux (définition donnée par l'ITU - *International Telecommunication*

Union). Il peut-être quantifié à la fois dans le domaine temporel et dans le domaine fréquentiel. Dans le domaine fréquentiel, on parle plutôt de bruit de phase (*phase noise*) : alors qu'un signal sinusoïdal idéal ne devrait présenter qu'une raie dans le domaine fréquentiel, on observe en réalité une dispersion des fréquences autour de cette raie. Plus on s'éloigne de la fréquence centrale du signal, plus celle-ci diminue jusqu'à atteindre le seuil du bruit blanc. Le bruit de phase est donc mesuré à une fréquence *offset* par rapport à la fréquence centrale du signal en db. Une métrique souvent utilisée est dbc/Hz (*db relative to the carrier per Hertz*). Dans cette section, nous nous intéressons principalement au *jitter* dans le domaine temporel.

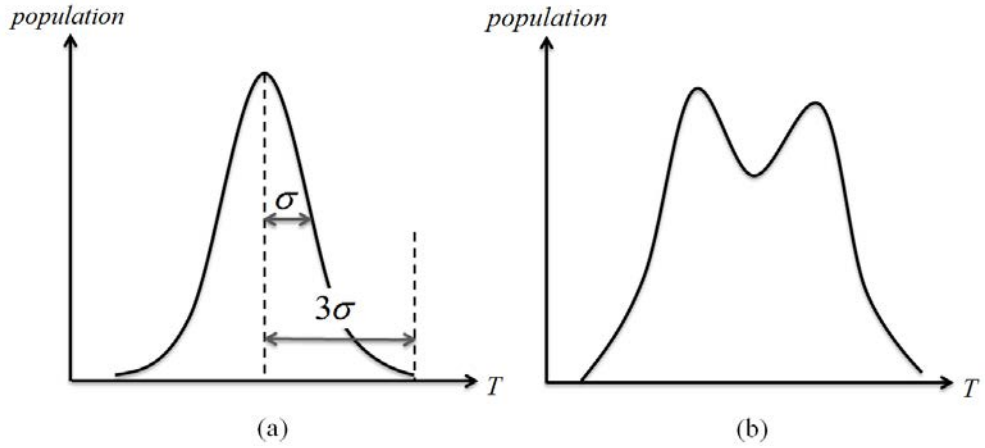


Figure 4.1 – Distribution des périodes d'un oscillateur : (a) *jitter* de type aléatoire (b) *jitter* de type aléatoire + une composante déterministe simple

Traditionnellement, le *jitter* est classifié selon deux catégories : aléatoire et déterministe. Le *jitter* observé en pratique dans un signal numérique est souvent un mélange des deux.

Composante aléatoire du *jitter* Elle provient du bruit intrinsèque dans les composants. Certains types de bruit, comme le bruit thermique (mouvements aléatoires des porteurs de charges), sont présents dans tous les composants électroniques (il est impossible de s'en affranchir). Le bruit thermique est généralement assimilé à un bruit blanc avec un spectre plat et possède des propriétés parfaitement aléatoires. Le *flicker* est un autre type de bruit aléatoire omniprésent dans les circuit. Il est aussi appelé bruit en $1/F$, bruit de scintillation, ou bruit rose (*pink noise*) car son amplitude diminue linéairement avec la fréquence (dans la densité spectrale d'un signal) jusqu'à atteindre le seuil du bruit blanc. Ce bruit concerne donc principalement les basses fréquences. Son origine n'est pas clairement établie (c'est un phénomène qu'on retrouve également dans des systèmes mécaniques comme les fluctuations de la période de rotation de la terre et les mouvements de courants maritimes) mais elle semble liée à la superposition de processus de relaxation [1] (*relaxation processes*). Dans le cadre des circuits numériques, les bruits aléatoires se répercutent sur les temps de propagation des opérateurs de base (portes logiques) selon une distribution statistique qui suit le plus souvent une loi normale (partie (a) de Fig. 4.1). Celle-ci peut être quantifiée par sa moyenne

1. http ://arxiv.org/ftp/physics/papers/0204/0204033.pdf

μ et son écart-type σ. L'écart-type σ est alors interprété comme l'amplitude du *jitter*. La partie aléatoire du *jitter* n'est par ailleurs pas bornée.

Composante déterministe du *jitter* Elle correspond à des erreurs de *timing* des signaux manipulés, celles-ci proviennent de divers facteurs et ne sont pas de nature aléatoire. Elles peuvent être périodiques (par exemple l'influence d'une alimentation à découpage) ou liées aux données et à la nature du système. Un exemple typique de *jitter* déterministe peut être observé dans un signal numérique en modulant sa tension d'alimentation avec un signal périodique (partie (b) de Fig. 4.1).

La thèse présentée dans [Val10] propose une caractérisation exhaustive du *jitter* dans différents type d'oscillateurs numériques (mis à part les anneaux auto-séquencés). L'auteur note que, dans des conditions idéales (qualité du circuit et surtout de l'alimentation), les anneaux à base d'inverseurs présentent un *jitter* aléatoire dû aux composantes de bruit blanc et $1/F$. Cependant, dans les basses fréquences (quand on augmente le nombre d'étages), la part de *flicker* devient rapidement dominante. Par contre, les oscillateurs à quartz présentent souvent une composante déterministe périodique et des distributions statistiques comportant des défauts. Les DFS (*Digital Frequency Synthesis*) présentent quant à elles un *jitter* largement dominé par les composantes déterministes. Le signal en sortie d'un PLL (*Phase-locked Loop*) présente généralement un profil aléatoire, mais il peut aussi contenir des composantes périodiques en fonction des paramètres réglés.

Dans le cadre des oscillateurs numériques, une métrique souvent utilisée pour quantifier le *jitter* est l'écart-type de leurs périodes d'oscillation (on parle de *period jitter*), noté σ_{period}. Cependant, celui-ci n'a une valeur qu'on peut interpréter que dans le cas de distributions gaussiennes. Pour les anneaux à inverseurs, le *period jitter* est dû à l'accumulation du *jitter* dans chaque étage de l'anneau. Dans [FBBV08], les auteurs modélisent chaque étage d'un anneau à inverseurs comme une source de bruit indépendante (leur délai de propagation est modélisé par une loi normale de moyenne D et d'écart-type σ_{porte}). Ensuite ils établissent le lien entre l'écart-type de la période d'oscillation σ_{period} et σ_{porte} en fonction du nombre d'étages k de l'anneau (en supposant que le bruit est indépendant de la fréquence) :

$$\sigma_{period} = \sqrt{2k}\sigma_{porte} \tag{4.1}$$

3. Simulations numériques

Dans cette section, nous présentons, analysons et simulons un modèle VHDL des anneaux auto-séquencés qui intègre à la fois le *jitter* ainsi que les effets temporels propres à la porte de Muller (effets Charlie et de drafting). L'un des principaux objectifs de ce modèle est d'étudier le lien entre le *jitter* généré localement dans un étage de l'anneau et celui que l'on retrouve sur la période d'oscillation.

3.1. Modélisation haut-niveau des anneaux et du *jitter*

Le schéma de la simulation est illustré dans Fig. 4.2. L'implantation VHDL des anneaux constitue le coeur de cette simulation. Les librairies TAL (*Tima Asynchronous Library*) ont été conçues au laboratoire TIMA afin de proposer les outils et modèles pour la simulation numérique de circuits asynchrones. Nous utilisons le *package charlie_function*. Celui-ci permet de stocker dans des variables les temps des différents événements en entrée (Ta et Tb) et en sortie (Tz) d'une porte de Muller, puis de calculer le délai de propagation selon l'équation *Charlie* (Eq. (2.2)).

La librairie RNG a été développée par Gnanasekaran Swaminathan [2]. Elle permet de générer des distributions statistiques selon plusieurs types de lois de probabilités (loi normale, loi de Poisson, loi binomiale ...). Nous nous servons de cette librairie pour générer des petites variations temporelles autour des délais de propagation de chaque étage selon une loi normale avec un écart-type paramétrable. A chaque fois qu'un délai de propagation est évalué par le simulateur, il est fait appel à la fonction *get.rnd()* qui génère une nouvelle réalisation issue d'une loi normale qu'on a paramétré. Afin que les réalisations des différents étages ne soient pas inter-dépendantes, chaque étage génère sa propre loi de probabilité avec des graines d'initialisation différentes.

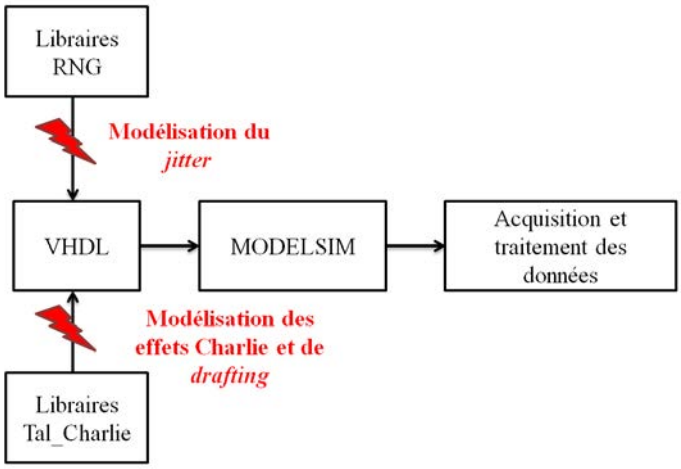


Figure 4.2 – Schéma de principe de la simulation

Pour chaque étage d'anneau auto-séquencé, nous pouvons donc régler les paramètres suivants :

- D_{ff} et D_{rr} les délais de propagation statiques en ps
- $D_{charlie}$ l'amplitude de l'effet Charlie en ps
- A_d et B_d la durée (en ps) et l'amplitude (en ps) de l'effet de drafting
- *seed* la graine d'initialisation pour la génération de la loi normale
- σ l'écart-type de la loi normale

A des fins de comparaison, nous avons également implanté des anneaux à inverseurs, leur délai de propagation suit un modèle linéaire standard, les paramètres de chaque étage sont les suivants :

2. http ://www.ittc.ku.edu/EECS/EECS_546/magic/files/vlsi/vhdl/random.pkg

- D le délai de propagation en ps
- *seed* la graine d'initialisation pour la génération de la loi normale
- σ l'écart-type de la loi normale

A l'issue de la simulation numérique sous Modelsim, les signaux sont acquis sous format texte, le traitement des données est ensuite effectué grâce à un programme C que nous avons réalisé. Enfin l'affichage se fait grâce à des scripts sous R. Dans la suite et pour toutes les expériences, nous fixons $D_{ff} = Drr = D = 500$ ps (environ la valeur constatée sur une technologie CMOS 350 nm dans nos implantations dans le chapitre 4). A moins d'être explicités, $D_{charlie}$ vaut 100 ps (proche de la valeur constatée dans nos expériences), A_d vaut 50 ps et B_d vaut 50 ps. Nous avons remarqué que ces deux dernières valeurs (A_d et B_d) influent assez peu sur les résultats des expériences lorsque le mode d'oscillation est régulier (dans le mode régulier c'est l'influence de l'effet *Charlie* qui est déterminante). Enfin, l'écart-type de la loi normale est fixé à $\sigma = 10$ ps. Cette valeur est un peu grande par rapport à la réalité mais elle permet de profiter d'une meilleur précision numérique pour la simulation.

3.2. Propagation des variations temporelles dans un anneau auto-séquencé

Notre objectif dans un premier temps est d'observer la propagation du *jitter* dans les anneaux auto-séquencés. En particulier, nous souhaitons quantifier l'influence du bruit local d'un étage donné sur les autres étages. Pour cela, nous proposons l'expérience illustrée dans Fig. 4.3, qui implique un anneau auto-séquencé (STR) et un anneau à inverseurs (IRO), tous deux à 64 étages. L'anneau STR64_32J est initialisé avec 32 jetons alors que STR64_26J est initialisé avec 26 jetons. Un étage (indexé 0) agit comme une source de bruit : il génère des variations temporelles autour de son délai de propagation avec un écart-type $\sigma = 10$ ps. Les autres étages ont un comportement idéal, ils ne génèrent pas de *jitter*.

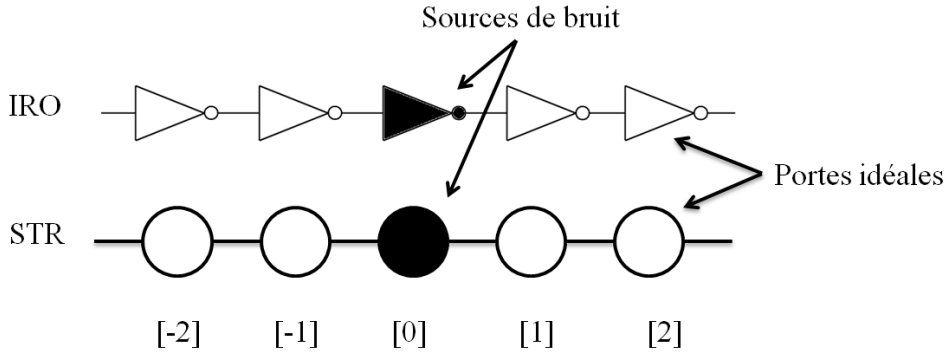


Figure 4.3 – Principe de l'expérience et numérotation des étages

Dans Tab. 4.1, nous mesurons l'écart-type de la période d'oscillation à la sortie des différents étages des anneaux IRO64, STR64_32J et STR64_26J. Le but est d'évaluer à quel point on

retrouve les variations produites par l'étage 0 dans les autres étages en fonction de leur éloignement. Notons que STR64_32J fonctionne avec un effet *Charlie* maximal car les temps de séparation sont nuls ($s = 0$) (le point de fonctionnement est dans la "vallée" de la courbe *Charlie*), ce qui n'est pas le cas de STR64_26J (le point de fonctionnement est proche de la zone linéaire de la courbe *Charlie*).

Etage	-16	-8	-3	-1	0	1	3	8	16
IRO64	13.88	13.88	13.88	13.88	13.88	13.88	13.88	13.88	13.88
STR64_32J	0.89	0.89	2.08	6.56	12.83	6.58	2.08	0.89	0.89
STR64_26J	0.98	0.78	0.74	1.45	13.13	12.08	9.82	6.53	3.82

Table 4.1 – Déviation standard de la période d'oscillation (en ps) mesurée à la sortie de différents étages des anneaux

Dans IRO64, les variations temporelles introduites dans l'étages 0 sont mesurées de manière égale dans tous les étages $\sigma_{period} = 13.88$ ps $\simeq \sqrt{2 \times 1}\sigma$. Nous mesurons également cette même valeur si on change le nombre d'étages (tant qu'un seul étage agit comme source de bruit). Ceci est expliqué par le fait qu'un événement se propage dans l'anneau sans contrainte. S'il prend un retard ou une avance δ par rapport à ses temps idéaux il conservera cet écart n étages plus loin. La propagation d'un événement dans un anneau à inverseurs se fait donc sans perte d'information : le *jitter* généré dans un étage influe sur le signal quelque soit l'étage où on le mesure.

Dans STR64_32J, il apparaît clairement que les variations temporelles introduites dans l'étage 0 sont fortement atténuées, et de manière symétrique, dans les étages adjacents ($\sigma_{period} = 6.56$ ps pour l'étage 1 et $\sigma_{period} = 2.08$ ps pour l'étage 3) jusqu'à atteindre un seuil minimal de $\sigma_{period} = 0.89$ ps. Ce dernier semble lié à la précision du simulateur (il correspond à une population composée de deux valeurs de périodes espacées de 1 ps). Ceci s'explique par le fait que la propagation des événements est contrainte par les effets *Charlie* et de *drafting* dans l'anneau auto-séquencé : celui-ci régule les temps entre événements au fur et à mesure qu'il s'y propagent. L'effet est d'autant plus important que le point de fonctionnement est proche de la vallée de la courbe *Charlie* (*i.e.* $s = 0$ quand $D_{ff} = D_{rr}$)). Plus précisément, comme on peut le voir dans Fig. 2.16, la courbe présente un profil presque plat dans cette zone : des variations temporelles autour de s sont réduites dans $charlie(s)$. Par exemple, en supposant $D_{ff} = D_{rr}$ ($s_{min} = 0$ dans Eq. (2.2)), alors :

$$\frac{\partial charlie(s,y)}{\partial s} = \frac{s}{\sqrt{s^2 + {D_{charlie}}^2}} \tag{4.2}$$

Plus $D_{charlie}$ est grand et plus s est proche de 0, plus les variations autour de s sont réduites dans $charlie(s, y)$.

En ce qui concerne STR64_26J, cette régulation des variations temporelles est beaucoup moins marquée. En effet, le point de fonctionnement de cette configuration est loin de $s = 0$, on s'approche donc d'un comportement linéaire. Il est intéressant de remarquer que la réduction des variations temporelles n'est pas symétrique ($\sigma_{period} = 12.08$ ps pour l'étage 1 et $\sigma_{period} = 1.45$ ps pour l'étage -1). En fait, comme l'anneau fonctionne dans la zone limitée en jetons, les signaux d'acquittement des étages sont déjà établis lors de l'arrivée du signal de requête (présence d'un jeton en entrée

de l'étage), le temps de basculement d'un étage dépend donc plus du précédent que du suivant (le temps de séparation entre les deux n'influe pas beaucoup car il est très grand).

Pour conclure, cette expérience montre que la propagation des événements dans un anneau auto-séquencé se fait avec perte d'information : le *jitter* généré dans un étage influe de moins en moins sur la période d'oscillation à mesure qu'on s'éloigne de l'étage référence (source de bruit). Ceci est particulièrement important dans le cadre du STRNG car cela veut dire que le *jitter* mesuré à la sortie d'un étage correspond principalement à celui généré dans cet étage même (et dans la moindre mesure celui généré dans les deux étages adjacents) et que les dépendances entre bits échantillonnés peuvent être raisonnablement négligées sous certaines conditions lors de la construction du modèle stochastique.

3.3. Analyse de la variance de la période d'oscillation

Dans les expériences suivantes, chaque étage des anneaux agit comme une source de bruit indépendante (l'écart-type σ est le même pour tous les étages mais les graines d'initialisation sont différentes). Dans un premier temps, nous vérifions simplement que les effets *Charlie* et de *drafting* n'introduisent pas de composante déterministe dans la période d'oscillation des anneaux auto-séquencés. Fig. 4.4 montre la distribution des périodes d'un anneau auto-séquencé à 63 étages initialisé avec 32 jetons (configuration multiphases) dont tous les étages agissent comme sources de bruit indépendantes. On retrouve une distribution gaussienne (ce qui nous permet d'interpréter correctement les écart-types sur la période d'oscillation σ_p dans la suite).

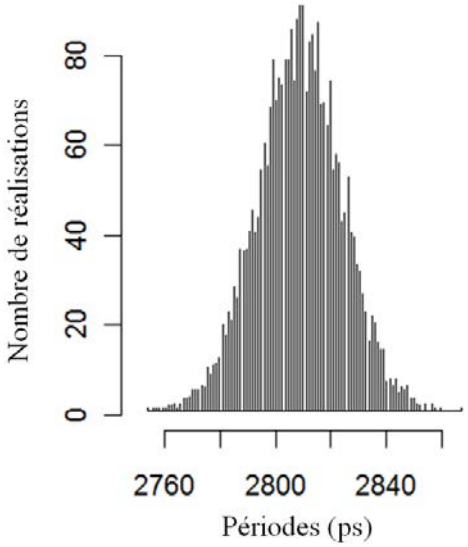


Figure 4.4 – Distribution des périodes d'un anneau auto-séquencé à 63 étages initialisé avec 32 jetons

Nous considérons d'abord un anneau auto-séquencé avec $N_T = N_B$ (il fonctionne donc à sa fréquence maximale), nous souhaitons établir si le nombre d'étages influe sur le *period jitter*

d'un anneau auto-séquencé. Nous mesurons donc σ_p (écart-type de la population constituée des périodes d'oscillations successives) et σ_{hp} (écart-type de la population constituée des demi-périodes d'oscillations successives) en fonction du nombre d'étages de l'anneau. Ces mesures sont présentées dans Fig. 4.5. Comme on peut le voir, on mesure une valeur $\sigma_p \simeq 17.5$ ps $(= 1.75\sigma)$ constante quelque soit le nombre d'étages. Il est très intéressant de noter qu'elle est comprise entre $\sqrt{2}2\sigma$ qui est la valeur théorique pour un anneau à inverseurs composé d'un seul étage selon Eq. (4.1) et 2σ qui est la valeur théorique pour anneau à inverseurs à 2 étages. Enfin, dans un cas plus général $(N_T \neq N_B)$, nous avons constaté que σ_p ne variait pas avec le nombre d'étages L du moment que l'on conserve le même rapport N_T/L, ce qui suggère que σ_p dépend du temps de séparation s en régime permanent (celui-ci dépend du taux d'occupation de l'anneau).

Cette fois, on effectue la même mesure sur un anneau auto-séquencé de 32 étages en faisant varier son taux d'occupation (nombre de jetons). Les résultats sont tracés dans Fig. 4.6. Cette courbe démontre clairement la dépendance entre l'amplitude du *period jitter* (σ_p) et le temps de séparation en régime permanent s (déterminé par le taux d'occupation de l'anneau). Ce résultat est compréhensible au vu de l'analyse de la section précédente, en particulier si on compare les configurations STR64_26J et STR64_32J de Tab. 4.1. On peut remarquer que pour $N_T = 12$ (un jeton pour trois étages), on trouve $\sigma_p = 31$ ps $\simeq \sqrt{2 \times 5}\sigma$ qui est la valeur théorique pour un anneau à inverseurs de 5 étages.

Généralement, les paramètres des effets *Charlie* et de *drafting* ne sont pas forcément connus dans un circuit donné. Dans nos expériences, nous avons constaté que l'effet de *drafting* avait une influence significative sur le mode de propagation en rafale des anneaux auto-séquencés ainsi que sur la gamme de jetons qui donnait lieu à une propagation régulière. Cependant, il influe très peu sur les périodes d'oscillation dans le mode régulier. Nous avons remarqué que c'était aussi le cas pour les valeurs de *jitter*. Cependant, l'influence de l'effet *Charlie* est un peu plus notable (à la fois sur les périodes d'oscillation et sur le *jitter*). Dans Tab. 4.2 nous mesurons σ_p pour un anneau auto-séquencé avec $N_T = N_B$ (dans la zone de fonctionnement où l'effet *Charlie* est maximal), pour différentes valeurs de l'amplitude de l'effet *Charlie* $D_{charlie}$. On voit que plus cette dernière est grande, plus σ_p a une valeur petite, cependant elle semble converger assez rapidement vers une valeur proche de 1.7σ.

$D_{charlie}$	σ_{hp} (ps)	σ_p (ps)
10	15.01	18.83
100	14.41	17.12
200	14.04	16.91
1000	13.95	16.88

Table 4.2 – Déviation standard de la période (σ_p) et de la demi-période (σ_{hp}) d'un anneau auto-séquencé de 32 étages en fonction de l'amplitude de l'effet Charlie

Pour terminer, étant donné le contexte d'utilisation des anneaux auto-séquencés dans le STRNG, nous effectuons des mesures sur des anneaux dans leur configuration multiphasée (N_T et L sont premiers entre eux). Ces mesures sont présentées dans Tab. 4.3. En augmentant le nombre d'étages de l'anneau, on améliore sa résolution de phase $(\triangle\varphi_{moy})$. De plus, le temps de séparation en régime

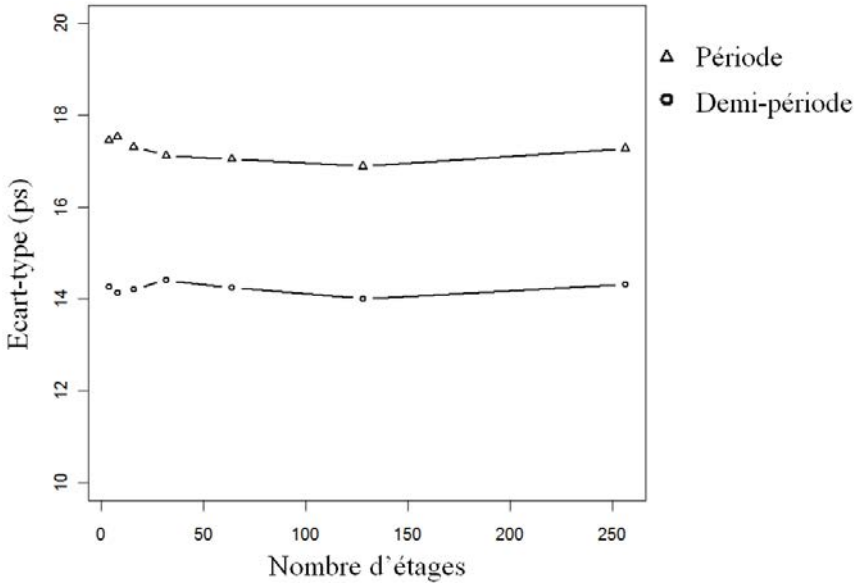


Figure 4.5 – Déviation standard de la période (σ_p) et de la demi-période (σ_{hp}) d'un anneau auto-séquencé avec $N_T = N_B$ en fonction de son nombre d'étages

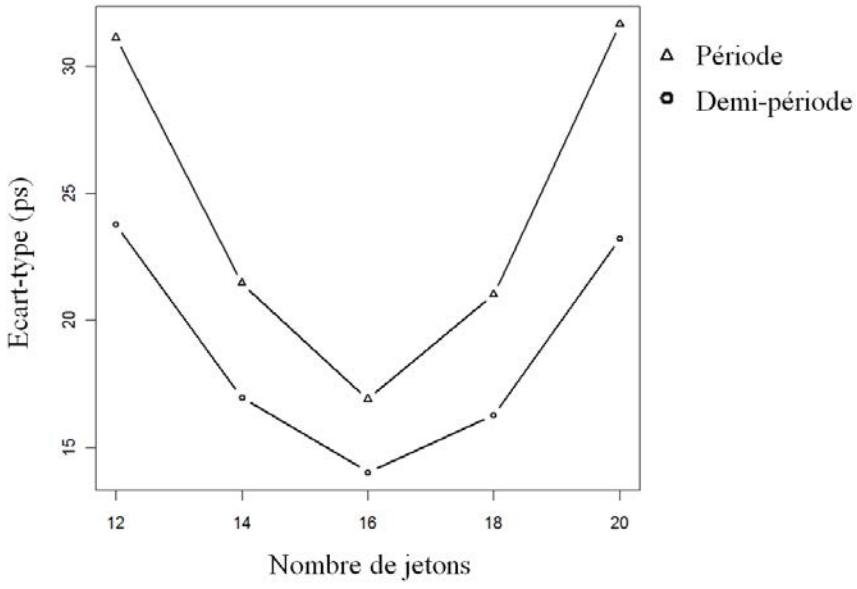


Figure 4.6 – Déviation standard de la période (σ_p) et de la demi-période (σ_{hp}) d'un anneau auto-séquencé de 32 étages en fonction de son taux d'occupation

permanent s devient de plus en plus petit, et l'amplitude du *period jitter* converge vers celle d'un anneau dans sa configuration optimale ($N_T = N_B$ avec $\sigma_p = 1.7\sigma$).

L	N_T	$\triangle\varphi_{moy}$ (ps)	σ_{period} (ps)
5	4	525	25.68
15	8	188	21.30
31	16	90	17.48
63	32	44	16.99
127	64	22	16.93

Table 4.3 – Résolution de phase moyenne ($\triangle\varphi_{moy}$) et écart-type de la période (σ_{period}) pour les configurations multiphasées (pour $D_{charlie} = 200$ ps)

3.4. Synthèse des résultats

Dans les oscillateurs à inverseurs classiques, le *jitter* mesuré sur le signal de sortie est dû à l'accumulation des variations temporelles que connait un événement à chaque fois qu'il traverse un étage de l'anneau. Ces variations s'ajoutent de manière linéaire (les écarts-type s'ajoutent en racine). Par conséquent les temps de basculement d'un étage de l'anneau dépendent à la fois du bruit de l'étage concerné, mais aussi de celui de tous les autres étages qui constituent l'anneau.

A contrario, le *jitter* mesuré sur le signal de sortie d'un anneau auto-séquencé avec un taux d'occupation optimal (selon Eq. (3.1)) est principalement dû aux variations temporelles que connaît un événement quand il traverse l'étage concerné (il dépend peu des variations qu'il a pu subir dans les étages précédents). Par conséquent les temps de basculement d'un étage dépendent surtout de son bruit local, et dans la moindre mesure de celui des étages adjacents (l'influence diminue en s'éloignant de l'étage concerné).

Dans le cas général, le *jitter* dans les anneaux auto-séquencés dépend essentiellement de trois paramètres : le niveau de bruit local dans chaque cellule (représenté par σ l'écart-type du temps de propagation de la cellule), l'amplitude de l'effet *Charlie* $D_{charlie}$ et le temps de séparation s dans le régime permanent (qui dépend du taux d'occupation de l'anneau). Plus $D_{charlie}$ est grand, et plus s est petit, plus l'écart-type de la période d'oscillation σ_p est petit. Celui-ci converge alors vers une valeur proche de 1.7σ.

Cependant, il faut préciser que la limite de cette simulation tient du fait que la modélisation ne prend pas en compte la dépendance fréquentielle du *jitter* (en effet on suppose son amplitude constante quelque soit la fréquence comme pour un bruit blanc). Cela est par contre compensé par le fait que les anneaux auto-séquencés fonctionnent à des fréquences généralement hautes où l'influence des bruits de *flicker* est assez faible.

4. Anneaux auto-séquencés dans des cibles FPGA

Dans cette section, nous implantons et caractérisons des anneaux auto-séquencés dans deux cibles FPGA : Altera Cyclone III et Xilinx Virtex 5. En particulier, nous mesurons leur période d'oscillation et leur amplitude de *jitter*, ensuite nous évaluons leur robustesse aux variations de tension.

4.1. Implantation des anneaux

Nous avons constaté à travers notre expérience que l'implantation d'anneaux auto-séquencés dans les FPGA pose un certain nombre de problèmes et nécessite d'avoir accès aux ressources bas-niveau (définition manuelle du contenu des LUT, accès aux ressources de placement/routage ...). D'autre part, les outils de synthèse commerciaux ne sont pas adaptés à l'implantation de circuits asynchrones car ils opèrent des simplifications sur les blocs combinatoires entre les registres de mémorisation. Cependant, nous montrons que l'implantation d'anneaux auto-séquencés dans des FPGA est faisable si on respecte un certain nombre de contraintes. Nous proposons ici une méthode d'implantation, *a priori* valable pour les technologies FPGA à base de LUT disposant d'au moins 4 entrées et à condition d'avoir accès aux ressources de placement/routage.

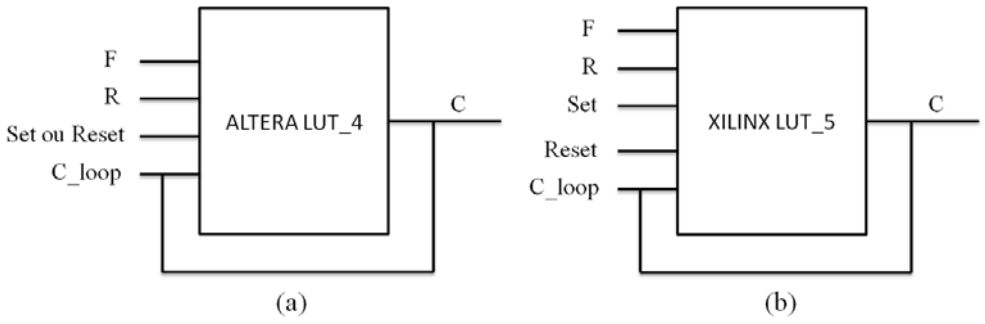


Figure 4.7 – Architecture d'un étage de l'anneau dans : (a) Altera Cyclone III (b) Xilinx Virtex 5

La fonction logique qui exprime la sortie d'un étage d'anneau auto-séquencé C en fonction de ses entrées F et R et de sa sortie mémorisée C_{loop} est donnée par l'équation ($\cdot$ désigne l'opérateur ET et $+$ désigne l'opérateur OU) :

$$C = F \cdot \overline{R} + F \cdot C_{loop} + \overline{R} \cdot C_{loop} \tag{4.3}$$

On peut rajouter des signaux d'initialisation *set* et rst de manière assez simple (ici *set* est prioritaire) :

$$C = (F \cdot \overline{R} + F \cdot C_{loop} + \overline{R} \cdot C_{loop}) \cdot \overline{rst} + set \tag{4.4}$$

Cependant, comme le fonctionnement des pipelines asynchrones n'est pas séquencé par une horloge globale, leurs cellules de base ne doivent pas présenter d'aléas logiques (*i.e* que la sortie ne

doit pas varier plusieurs fois avant d'atteindre sa valeur finale). Il faut alors contraindre cette fonction logique dans une seule cellule de base du FPGA (elle ne doit pas être réalisée avec des portes séparées). Il faut donc créer un composant VHDL qui instancie une LUT (cellule logique programmable) et programme son contenu. Par exemple, cela peut être fait dans Xilinx Virtex 5 en initialisant une LUT avec le contenu "0000FFB2" [3], ce qui réalise la fonction décrite par Eq. (4.4) :

STR_stage : LUT5_D GENERIC MAP (INIT => X"0000FFB2") PORT MAP (...)

La fonction de mémorisation est obtenue en rebouclant la sortie de la LUT sur l'une de ses entrées C_{loop} comme cela est montré dans Fig. 4.7. Les LUT à 4 entrées ne permettent pas d'implanter à la fois les signaux *set* et rst, il faut donc utiliser des cellules différentes en fonction de la configuration que l'on souhaite initialiser (nombre de jetons). Il est possible de faire des étages d'anneaux auto-séquencés avec des LUT à 3 entrées, cependant cela implique d'ajouter un étage qui permet l'injection des jetons afin d'initialiser l'anneau. Or cela implique qu'un étage de l'anneau présente des caractéristiques temporelles très différentes des autres, et nous avons constaté que cela dégrade le fonctionnement de l'anneau (phénomène de *bottleneck* mis en évidence dans la section 4.4).

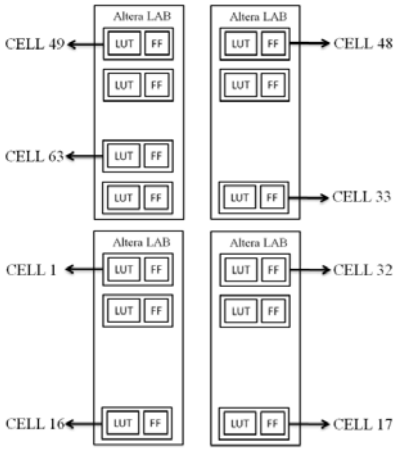


Figure 4.8 – Exemple de topologie pour un anneau à 63 étages dans Altera Cyclone III (chaque Altera Logic Block -LAB- contient 16 cellules logiques consistant en une LUT et une bascule)

Il est préférable d'avoir des anneaux équilibrés (les étages peuvent être asymétriques, mais doivent avoir des caractéristiques temporelles équivalentes) afin de maîtriser leurs modes et fréquences de fonctionnement et d'éviter la limitation de la fréquence de l'anneau par un étage beaucoup plus lent que les autres (phénomène de *bottleneck*). Dans beaucoup de FPGA commerciaux, les ressources de routage sont réparties de manière hiérarchique (ressources locales et globales), qui peut être irrégulière dépendant du placement dans le FPGA. Le placement et routage manuel

3. Les différentes combinaisons d'entrée correspondent aux adresses de la mémoire constituant la LUT, chaque adresse est associée à un bit du vecteur "0000FFB2"

des cellules de l'anneau sont nécessaires si on veut obtenir des caractéristiques temporelles similaires entre les différents étages. La topologie de l'anneau doit être choisie en prenant en compte les irrégularités de la structure de routage. Par exemple, pour Altera Cyclone III, la topologie la plus adaptée semble être un placement rectangulaire (comme montré dans Fig. 4.8), en utilisant les lignes locales intra LAB (*Altera Logic Block*) pour le rebouclage des cellules, et les lignes inter LAB (c'est des lignes qui longent chaque LAB) pour les connexions entre les étages.

4.2. Matériel et méthodes de mesure

Beaucoup de cartes d'évaluation fournies par les constructeurs de FPGA ne sont pas adaptées à la caractérisation et à la mesure du *jitter*. Les principaux défauts concernent souvent la mauvaise qualité de l'alimentation électrique. Par exemple, l'alimentation à découpage de la carte d'évaluation NIOS II D'Altera introduit des composantes déterministes importantes dans le profil du *jitter* mesuré [FBBV08].

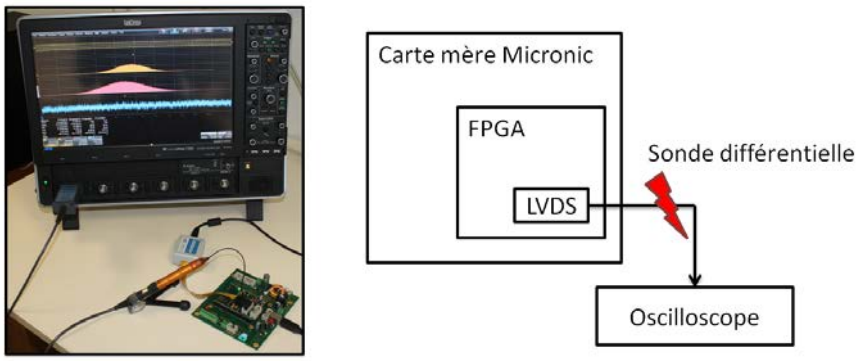


Figure 4.9 – Description du banc de mesure

L'équipe SES (*Secure Embedded Systems*) du laboratoire Hubert Curien a développé des cartes mères pour FPGA spécialement dédiées à la conception et l'évaluation de TRNG. Un soin particulier est apporté à la réalisation du circuit d'alimentation (utilisation d'un régulateur de tension linéaire) ainsi que les connectiques pour les mesures. Ces dernières utilisent la norme LVDS (*Low Voltage Differential Signaling*) pour la transmission de signaux électriques en fréquence élevée (typiquement de l'ordre de la centaine de MHz) sur une ligne symétrique. Les cartes mères comprennent également un module de transfert USB pour l'acquisition de données issues de sorties du FPGA (par exemple les bits de sortie d'un TRNG). L'équipe SES fournit et gère par ailleurs la plate-forme Evariste [FBH13], qui est un système modulaire multi-FPGA, open-source et basé sur ces cartes mères permettant le benchmarking en ligne de TRNG.

Pour effectuer nos mesures, nous disposons d'un oscilloscope à large bande passante *LeCroy Wavepro 735 ZI* (oscilloscope 4 canaux, 20 Gsample/s, 3.5 GHz de bande passante) fourni avec un ensemble d'outils pour l'analyse et calcul statistique (*Lecroy statistical tools*). Les signaux sont

acquis via les broches LVDS en utilsant une sonde différentielle avec un bande passante de 4 Ghz (c.f. Fig. 4.9).

4.3. Mesure préliminaire : l'effet *Charlie*

Alors que l'effet de *drafting* semble naturellement présent dans toutes les portes logiques (dû à la charge et décharge de capacité de sortie parasite), nous ne savons pas si une LUT programmée en cellule de Muller est aussi sujette à l'effet *Charlie* sur ses deux entrées et si les modèles précédents s'appliquent. Cependant, on peut supposer son existence du fait du comportement temporel des anneaux auto-séquencés implantés (présence des deux modes d'oscillation, en particulier le mode régulier).

Dans cette section, nous proposons un moyen, peu précis, mais qui permet de définir l'échelle de grandeur de l'effet *Charlie* sur un FPGA Altera Cyclone III. Le principe se base sur l'hypothèse que le délai de propagation d'une LUT (qui n'est rien d'autre qu'une "mémoire" programmable) ne dépend pas de la fonction qui y est implantée (*i.e* du contenu de la mémoire) et sur le fait que les délais par rapport aux différentes entrées sont les mêmes (ceux-ci sont à peu près égaux selon la notice technique et nos expériences). L'idée consiste alors à remplacer un étage d'un anneau à inverseurs (IRO) par un étage constitué d'une porte de Muller dont la sortie est inversée comme cela est montré dans Fig. 4.3 tout en gardant le même placement des cellules, puis de constater la différence de fréquence pour estimer l'effet *Charlie*. Nous appelons ce nouvel oscillateur MRO (*Muller gate based Ring Oscillator*). Les cellules sont placées et routées manuellement de manière à obtenir une structure symétrique (on utilise les lignes de routage locales dans un LAB).

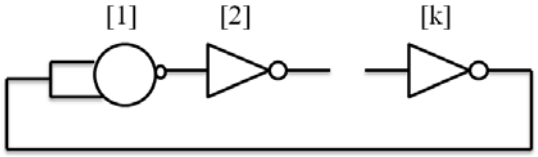


Figure 4.10 – Schéma de l'oscillateur MRO

L'anneau à inverseurs (oscillateur IRO) constitué de k éléments a une période T_{IRO} exprimée par Eq. (4.5) (D_{LUT} représente le délai de la LUT dans lequel est compris celui de l'interconnexion) :

$$T_{IRO} = 2kD_{LUT} \tag{4.5}$$

L'oscillateur MRO est placé et routé de manière identique que IRO sauf pour le premier étage comme le montre Fig. 4.3. La LUT du premier étage est programmée pour fournir la fonction d'une porte de Muller dont la sortie est inversée. La sortie de l'étage précédent est rebouclée sur ses deux entrées à la fois. Comme ces signaux arrivent quasi-simultanément sur les entrées de la LUT, si l'effet *Charlie* est effectivement présent, cet étage devrait présenter un délai plus important que les autres. En appliquant Eq. (2.2) avec $s = 0$ et $D_{ff} = D_{rr} = D_{LUT}$, et en négligeant l'effet de

drafting, alors cet étage a un délai $D_{LUT} + D_{charlie}$ où $D_{Charlie}$ est l'amplitude de l'effet *Charlie*. La période de l'oscillateur MRO est donc :

$$T_{MRO} = 2(k-1)D_{LUT} + 2(D_{LUT} + D_{charlie}) \tag{4.6}$$

Finalement, en implantant des oscillateurs MRO et IRO en faisant très attention à ce que leur placement et routage soit à peu près le même, il est possible d'obtenir une estimation grossière de l'effet *Charlie* en mesurant leurs périodes (D_{LUT} est exprimé en fonction de T_{IRO} selon Eq. (4.5)) :

$$D_{charlie} = \frac{T_{MRO} - \frac{k-1}{k}T_{IRO}}{2} - \frac{T_{IRO}}{2k} \tag{4.7}$$

Nous avons implanté des oscillateurs MRO et IRO de 3 et 5 étages et mesuré leur période d'oscillation. Dans nos différents tests, nous avons systématiquement observé une fréquence moins grande pour l'oscillateur MRO. En utilisant les périodes de MRO_3 et IRO_3 dans Tab. 4.4, on trouve $D_{LUT} = 261$ ps et $Dcharlie = 85$ ps. En utilisant les périodes de MRO_5 et IRO_5, on trouve $D_{LUT} = 272$ ps et $Dcharlie = 60$ ps. Ces mesures ne sont pas très précises mais elles donnent une idée de l'ordre de grandeur de l'effet *Charlie* dans les cellules à base de LUT.

Oscillateur	IRO_3	IRO_5	MRO_3	MRO_5
Période	1.57 ns	2.66 ns	1.74 ns	2.78 ns

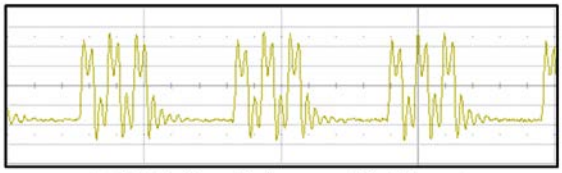
Table 4.4 – Périodes d'oscillation des oscillateurs IRO et MRO

4.4. Modes et fréquences d'oscillation

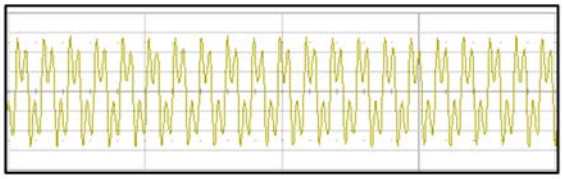
Les configurations implantées ont présenté un comportement similaire à celui attendu en théorie et observé dans les divers travaux publiés dans la littérature pour les implantations ASIC. A savoir, qu'on observe deux modes d'oscillations en fonction du taux d'occupation de l'anneau. Ceux-ci sont montrés dans Fig. 4.11. Pour un anneau à 64 étages, la gamme de jetons qui donne le mode régulier est $[24, 40]$ dans Altera et $[28, 38]$ dans Xilinx. Ces intervalles sont centrés autour de 33 pour Altera et 34 pour Xilinx (le nombre de jetons est en réalité toujours pair). Ceci laisse à penser que globalement $D_{ff} \simeq D_{rr}$ pour les deux implantations.

Dans la section 4.1, nous avons insisté sur la nécessité de concevoir des étages qui ont des caractéristiques temporelles proches, mais que se passe t-il si un étage a un délai beaucoup plus long que les autres ? Fig. 4.12 représente la courbe de fréquence de trois configurations d'anneaux auto-séquencés de 64 étages en fonction de leur taux d'occupation. La configuration notée ALT est une implantation dans Altera dont le placement et routage ont été laissés libres à l'outil automatique. La configuration notée ALT_PR est une implantation dans Altera qui a été placée et routée manuellement. Enfin, la configuration notée XIL_PR est une implantation dans Xilinx qui a été placée et routée manuellement.

La configuration ALT présente une courbe de fréquence plate : il existe dans la structure un délai très long qui limite la fréquence de fonctionnement de l'anneau. Ce phénomène, appelé



(a) Mode d'oscillation en rafale (6 jetons)



(b) Mode d'oscillation régulier (14 jetons)

Figure 4.11 – Modes d'oscillation d'un anneau auto-séquencé de 32 étages dans Altera Cyclone III

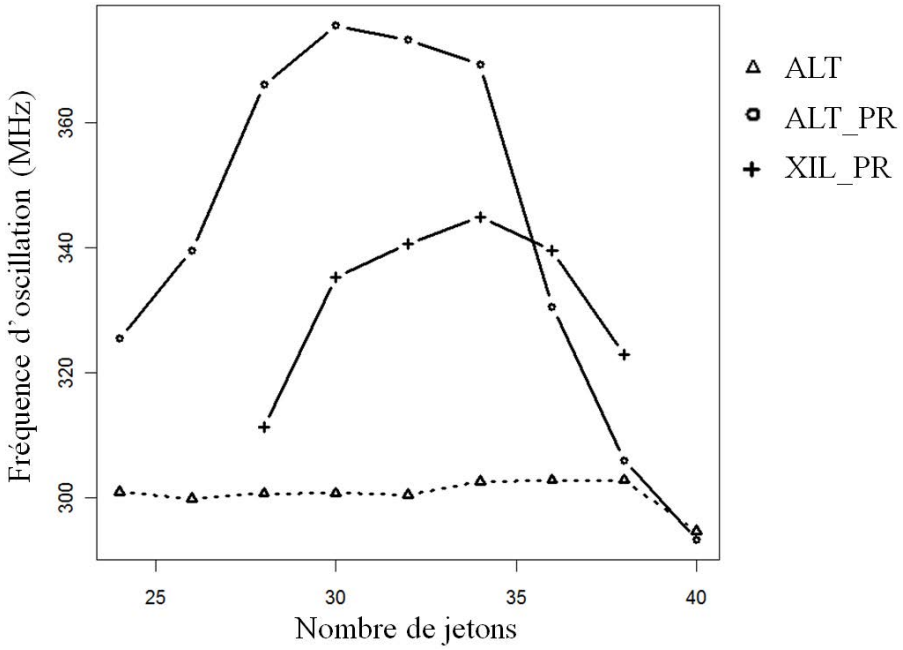


Figure 4.12 – Fréquences d'oscillations d'anneaux auto-séquencés à 64 étages en fonction de leur taux d'occupation dans Altera Cyclone III et Xilinx Virtex 5

bottleneck (goulot d'étranglement), est assez similaire au problème du même nom qu'on retrouve dans les chaînes de production : lorsqu'une partie d'une chaîne de traitement fonctionne moins vite que le reste, à long terme les matériaux traités peuvent s'accumuler à son entrée. Dans le contexte de l'anneau auto-séquencé, si un étage a un délai disproportionné par rapport aux autres, il ne pourra pas se synchroniser avec eux : les jetons qui arrivent à l'entrée de cet étage se retrouvent

séparés d'un temps très long qui limite la fréquence de fonctionnement de l'anneau. Par contre, les configurations qui ont été placées et routées montrent un profil de fréquence qui dépend du taux d'occupation de l'anneau, avec un maximum atteint autour de 30 jetons pour Altera et 34 jetons pour Xilinx.

4.5. Effets des variations de tension

Afin d'évaluer l'effet des variations de tensions sur les anneaux auto-séquencés, nous avons remplacé le régulateur de tension des cartes mères par un montage nous permettant de varier la tension d'alimentation du coeur du FPGA (Altera Cyclone III) entre 1 V et 1.4 V. Les configurations testées consistent en 2 anneaux à inverseurs à 5 et 80 étages (IRO_5 et IRO_80) et 3 anneaux auto-séquencés avec $N_T = N_B$ à 4, 24 et 96 étages (STR_4, STR_24 et STR_96). Nous mesurons leurs fréquences d'oscillation à différentes valeurs de la tension. Afin de comparer des configurations qui n'ont pas les mêmes fréquences, nous calculons les fréquences normalisées $F_n = F/F_{nom}$ où F est la fréquence relevée et F_{nom} est la fréquence nominale ($i.e$ à $1.2V$). Les courbes représentant les fréquences normalisées en fonction de la tension d'alimentation sont données dans Fig. 4.13.

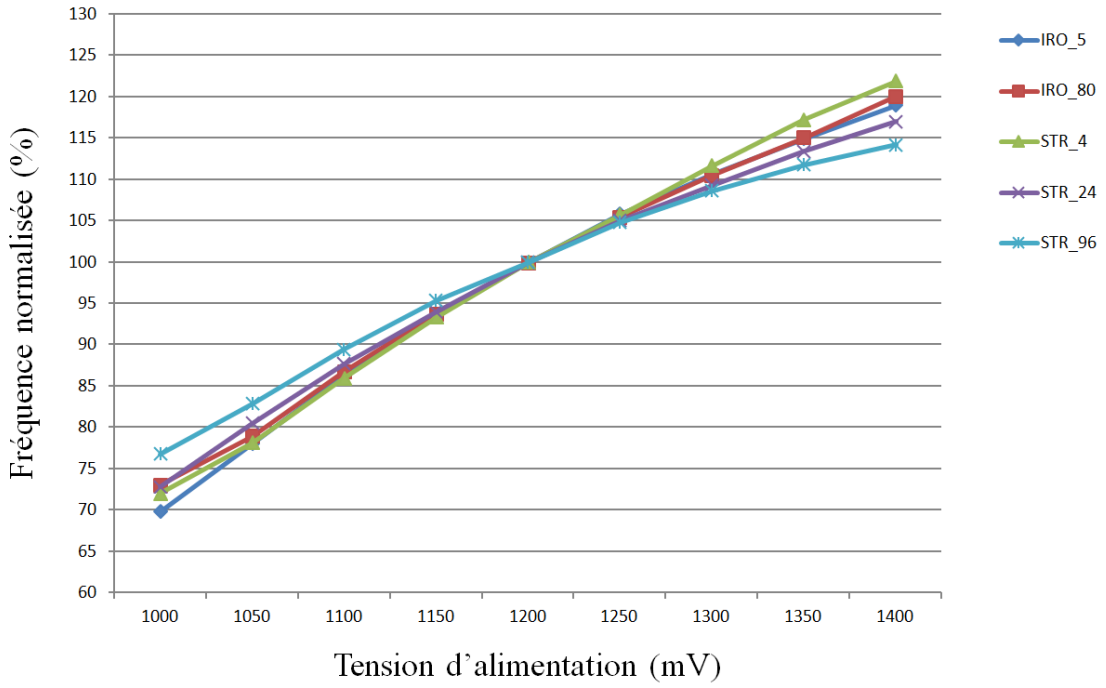


Figure 4.13 – Courbes des fréquences normalisées en fonction de la tension d'alimentation pour différents oscillateurs dans Altera Cyclone III

Les configurations STR_4, IRO_5 et IRO_80 ont des courbes très similaires (presque confondues) qui montrent une variation quasi-linéaire de la fréquence avec la tension d'alimentation. Par contre, STR_24 et STR_96 ont une courbe qui fait apparaître une non-linéarité aux extrémités.

Ces deux configurations sont aussi plus robustes aux variations de tension, c'est d'ailleurs plus marqué pour la configuration STR_96.

Le comportement non linaire des fréquences des anneaux auto-séquencés est prévisible du fait de la présence de l'effet *Charlie*, comme c'est le cas pour la courbe de fréquence en fonction du taux d'occupation. Ici encore, l'effet *Charlie* joue un rôle important dans la stabilité de la fréquence des anneaux auto-séquencés. Cependant, il semble qu'augmenter le nombre d'étages augmente la robustesse des anneaux auto-séquencés aux variations de tension (nous avons remarqué que c'était aussi le cas pour la robustesse à la variabilité du procédé de fabrication, même si cela n'est pas présenté ici). Cet aspect n'a pas été plus creusé dans le cadre de cette thèse, mais nous avons deux hypothèses pour expliquer cela :

- Notre analyse ne prend pas en compte des facteurs électriques comme la puissance du signal. En effet, en augmentant le nombre d'étages tout en conservant la même fréquence (ce qui est le cas quand $N_T = N_B$ et $D_{ff} = D_{rr}$), alors on augmente la puissance du signal et la consommation électrique [Iss10].
- En augmentant le nombre d'étages, on diminue la variabilité sur le rapport moyen D_{ff}/D_{rr}, qui statistiquement se rapproche de sa moyenne 1/2 (ce qui fait qu'avec $N_T = N_B$, le temps de séparation s se rapproche de plus en plus de 0).

4.6. Mesures de *jitter*

Toutes les configurations d'anneaux auto-séquencés qu'on a testées ont montré un profil de *jitter* de type gaussien avec des valeurs d'écart-type qui dépassent rarement les 5 ps. Fig. 4.14 montre la distribution des périodes d'un anneau auto-séquencé de 127 étages et 64 jetons dans Altera et Xilinx.

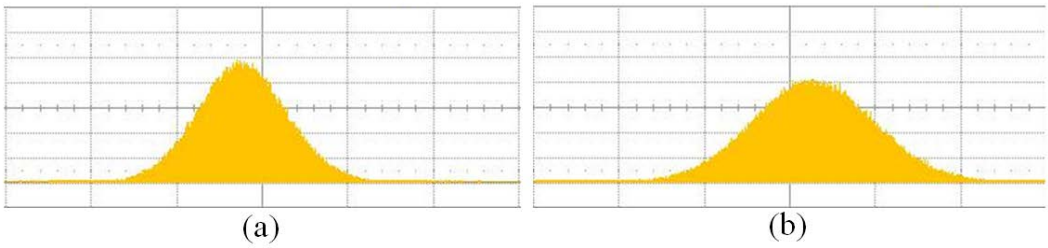


Figure 4.14 – Distribution des périodes d'un anneau auto-séquencé de 127 étages initialisé avec 64 jetons dans : (a) Altera Cyclone III (b) Xilinx Virtex 5 (l'échelle verticale est de 100 kilo-échantillon par division et l'échelle horizontale est de 5 ps par division)

Afin de nous situer par rapport à l'analyse présentée dans la section 3.3, nous commençons par estimer le *jitter* par cellule de base (*i.e.* l'écart-type des délais de propagation des étages). Comme le temps de propagation d'une mémoire ne dépend pas de son contenu (du moins si elle est correctement conçue), alors il est raisonnable de penser qu'une LUT programmée en inverseur génère le même niveau de bruit qu'une LUT programmée en étage d'anneau auto-séquencé. Nous

nous servons donc de configurations d'anneaux à inverseurs (dont le comportement est relativement connu) pour estimer l'écart-type du délai de propagation d'une LUT, noté σ_g. Fig. 4.15 représente l'écart-type des périodes d'oscillation d'un anneau σ_{period}, ainsi qu'une estimation de σ_g en utilisant une loi d'accumulation en racine donnée par Eq. (4.1).

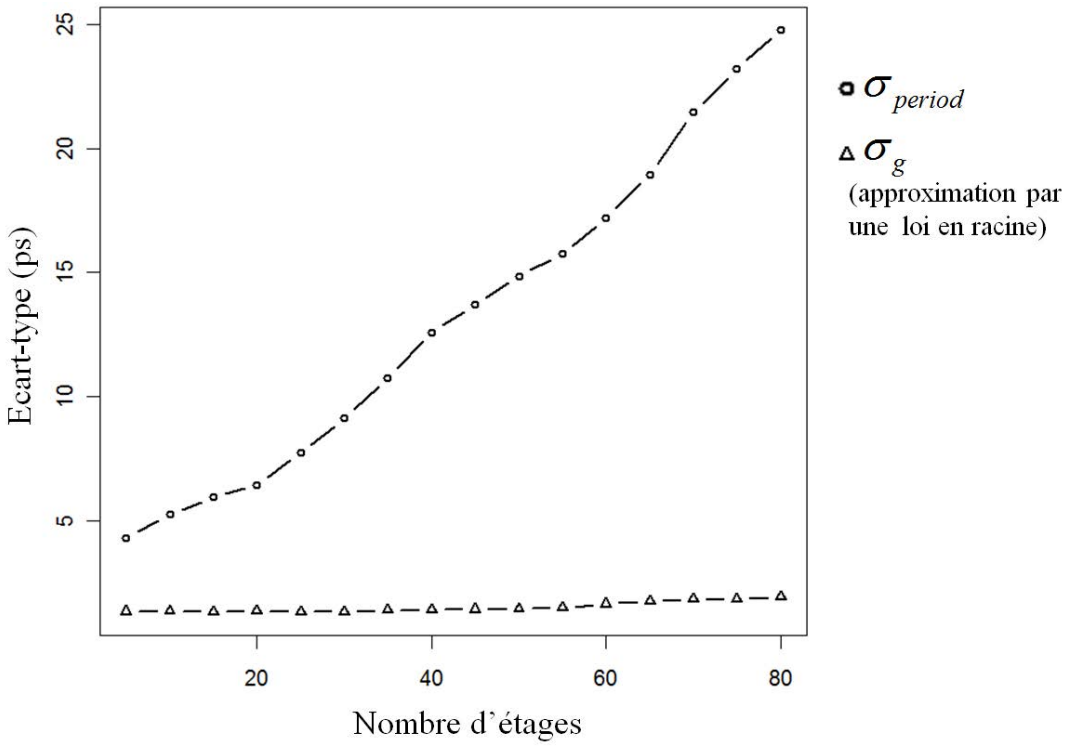


Figure 4.15 – Déviation standard de la période d'un anneau à inverseurs (σ_p) en fonction de son nombre d'étages et estimation du *jitter* par porte (σ_g) dans Altera Cyclone III

A partir d'une vingtaine d'étages, l'évolution de σ_p ne suit plus une courbe en racine. Ceci n'est pas étonnant à cause du *flicker noise* dont l'amplitude est plus grande dans les basses fréquences. Cele veut dire que l'approximation de loi en racine pour le calcul de σ_g est de moins en moins juste au fur et mesure qu'on augmente le nombre d'étages. D'autre part, la précision de la mesure de *jitter* est globalement moins bonne quand le nombre d'étages est petit. Néanmoins, on n'observe pas de différences significatives sur les valeurs de σ_g calculées au vu de la précision recherchée (la picoseconde). On trouve $\sigma_g = 1.4$ ps pour un faible nombre d'étages et $\sigma_g = 2$ ps pour un grand nombre d'étages (cette valeur est plus grande à cause du *flicker* qui est plus important à ces fréquences). Au vu des fréquences hautes des anneaux auto-séquencés, $\sigma_g \simeq 1.5$ ps est une estimation raisonnable pour le reste de l'étude. Précisons qu'au vu du matériel et des méthodes de mesures utilisées, une précision supérieure à la picoseconde est difficilement envisageable.

Dans le cas des anneaux auto-séquencés, comme attendu, l'amplitude du *jitter* ne dépend pas directement du nombre d'étages de l'anneau comme le montre Fig. 4.16. Les valeurs mesurées sont d'ailleurs très faibles et dépassent rarement 3 ps dans les cibles Altera (4.5 ps dans Xilinx). *A*

priori, selon les résultats de la section 3.3, on s'attend à ce qu'elles dépendent du taux d'occupation (ou plus précisément du temps de séparation s dans le régime permanent). Cependant, nous avons constaté que la variation par rapport au taux d'occupation était extrêmement faible dans ces implantations FPGA comme le montre Fig. 4.17. En réalité, il ne semble pas évident de dire si la précision de la mesure permet de discriminer des valeurs de *jitter* aussi petites. Il est d'ailleurs possible que la valeur mesurée pour les configurations optimales ($N_T = N_B$) soit plus grande que la valeur réelle.

En implantant des petites configurations (jusqu'à 16 étages) en utilisant les lignes de routage locales, on peut obtenir des anneaux auto-séquencés qui oscillent à des fréquences très rapides (autour de 600 MHz). Cependant, ces configurations affichent également des valeurs de *jitter* sensiblement proches de celles de configurations d'anneaux fonctionnant à des fréquences plus lentes (autour de 350 MHz), autour de 2.8 ps pour les cibles Altera. Ceci suggère que le bruit de *flicker* est faible pour les anneaux auto-séquencés (l'amplitude du bruit ne dépend pas de la fréquence).

4.7. Synthèse des résultats

Dans cette section, nous avons implanté des anneaux auto-séquencés et des anneaux à inverseurs classiques dans deux familles de FPGA afin de comparer leurs caractéristiques, en particulier celles qui nous intéressent dans le cadre de la génération d'aléa : les caractéristiques du *jitter* et la robustesse aux fluctuations externes (l'exemple étudié est celui des variations de tension).

La fréquence varie de manière quasi-linéaire avec la tension d'alimentation dans les anneaux à inverseurs quelque soit leur nombre d'étages. Par contre, dans les anneaux auto-séquencés, cette variation est de moins en moins linéaire et de plus en plus petite au fur et à mesure qu'on augmente le nombre d'étages (jusqu'à un certain seuil atteint autour d'une centaine d'étages).

En ce qui concerne le *jitter*, les mesures sur les anneaux à inverseurs montrent que l'écart-type des périodes d'oscillation augmente avec le nombre d'étages. Théoriquement, dans le cas d'un bruit blanc, il devrait augmenter suivant une loi en racine. Cependant, d'après nos mesures, il commence à augmenter rapidement de manière quasi-linéaire à partir de quelques étages (pas plus d'une vingtaine). Ceci est dû au fait que le bruit dans les basses fréquences est dominé par les bruits de *flicker* (bruits en $1/F$) : leur amplitude augmente plus la fréquence est basse. Néanmoins, ces mesures nous ont permis d'estimer une valeur pour l'écart-type du délai de propagation d'une LUT, noté σ_g, proche de 1.5 ps.

Les mesures sur les anneaux auto-séquencés implantés ont mis en évidence un *jitter* sur la période d'oscillation qui est très faible et du même ordre que celui généré localement dans les étages de l'anneau. Comme les fréquences d'oscillation sont généralement hautes (supérieures à 300 MHz dans les cibles Altera), l'influence du *flicker* semble plus réduite. Nous constatons des valeurs de *jitter* relativement constantes pour les différentes configurations, entre 2.5 ps et 3.5 ps dans Altera Cyclone III et entre 4 ps et 5 ps dans Xilinx Virtex 5.

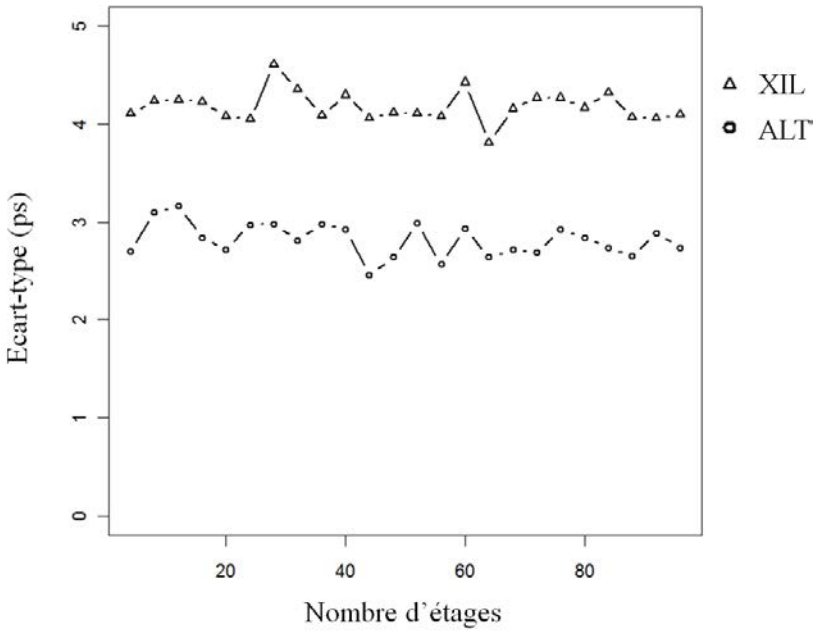


Figure 4.16 – Déviation standard de la période d'un anneau auto-séquencé avec $N_T = N_B$ en fonction de son nombre d'étages dans Altera Cyclone III (ALT) et Xilinx Virtex 5 (XIL)

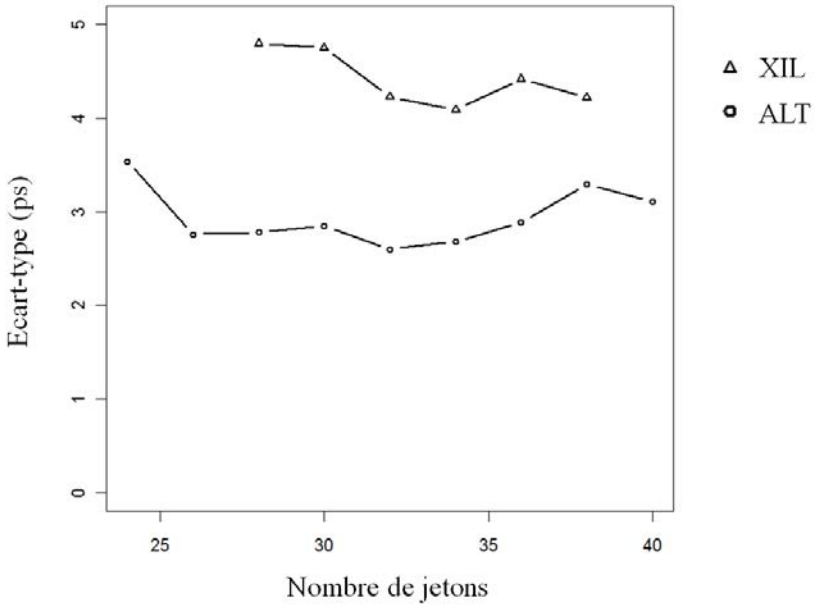


Figure 4.17 – Déviation standard de la période d'un anneau auto-séquencé de 64 étages en fonction de son taux d'occupation dans Altera avec (ALT_PR) ou sans placement/routage (ALT) et dans Xilinx (XIL)

5. Anneaux auto-séquencés dans une technologie CMOS 350nm

Un premier circuit CMOS a été réalisé dans le cadre de cette thèse dans une technologie AMS CMOS 350 nm. Le principal objectif de ce circuit est d'effectuer les mesures des périodes d'oscillation et amplitude de *jitter* nécessaires au dimensionnement du STRNG (*i.e.* le réglage de son taux d'entropie) pour une implantation concrète dans cette technologie.

5.1. Implantation des anneaux

Il existe différentes implantations des portes de Muller à base de transistors CMOS. Le travail de thèse présenté dans [Iss10] compare différentes implantations d'anneaux auto-séquencés dans une technologie STMicroelectronics CMOS 65 nm en fonction de leur fréquence, consommation et bruit de phase. Ce dernier critère nous intéresse particulièrement dans le cadre de cette thèse. Nous rappelons qu'un faible bruit de phase est, dans un cas général, plutôt un indicateur d'une bonne qualité de bruit (on se rapproche du palier bas lié au bruit blanc ayant un spectre plat). D'après ces travaux, les implantations de type dynamique et *weak feedback* (nécessitant au maximum 8 transistors) fonctionnent à des fréquences plus élevées que l'implantation dite conventionnelle (nécessitant 12 transistors), mais malgré cela, elles présentent un bruit de phase plus important.

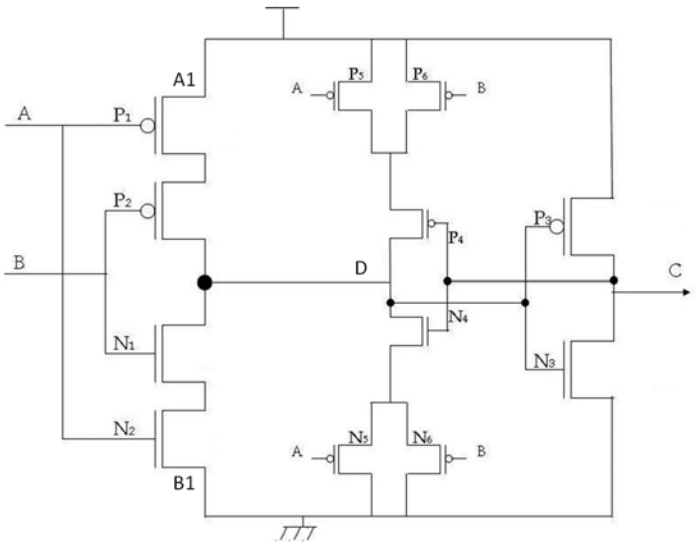


Figure 4.18 – Schéma transistor de la cellule de Muller conventionnelle

Nous avons choisi d'implanter les étages des anneaux auto-séquencés en utilisant l'implantation conventionnelle de la porte de Muller, représentée dans Fig. 4.18. Ce schéma est issu de l'implantation *weak feedback* représentée dans Fig. 2.5. Les transistors P_1, P_2, N_1 et N_2 servent à établir la valeur de sortie lorsque les deux entrées sont égales. Les transistors P_3, P_4, N_3 et N_4 constituent

deux inverseurs en tête-bêche qui servent à maintenir la valeur de la sortie quand les entrées sont différentes. Les transistors P_5, P_6, N_5 et N_6 (qui ne se trouvent pas dans l'implantation *weak feedback*) permettent d'éviter les conflits sur le noeud D en déconnectant l'inverseur de contre-réaction quand les entrées sont identiques. Enfin, nous avons rajouté 4 transistors permettant de connecter le point D soit à Vdd ou à gnd pour initialiser les étages (grâce à deux signaux set et $reset$), ceux-ci ne sont pas représentés dans Fig. 4.18.

Etant donné la finalité de ce circuit prototype (caractérisation des fréquences et du *jitter*), nous ne nous sommes pas imposés de contraintes vis-à-vis de la taille et consommation des cellules, leur *layout* n'est pas par ailleurs optimisé en surface. Nous avons donc dimensionné les transistors en utilisant la simulation électrique, en augmentant leur taille jusqu'à obtenir des pentes satisfaisantes des signaux. Le rapport entre les transistors du réseau P et ceux du réseau N a été sélectionné de manière à obtenir des temps de montée et des temps de descente égaux. Le *layout* a été réalisé de manière à simplifier la connectivité entre les étages et à permettre une topologie symétrique. Un *layout* d'un anneau auto-séquencé à 8 étages est représenté dans Fig. 4.19. Chaque étage possède quatre connexions, dont deux correspondant au même signal (sortie de l'étage) mais avec deux points d'accès en haut à droite et en bas à gauche des cellules. Par ailleurs, les étages aux extrémités ont un routage légèrement ajusté afin de conserver les mêmes propriétés temporelles tout le long de l'anneau (notamment grâce à la topologie en rectangle).

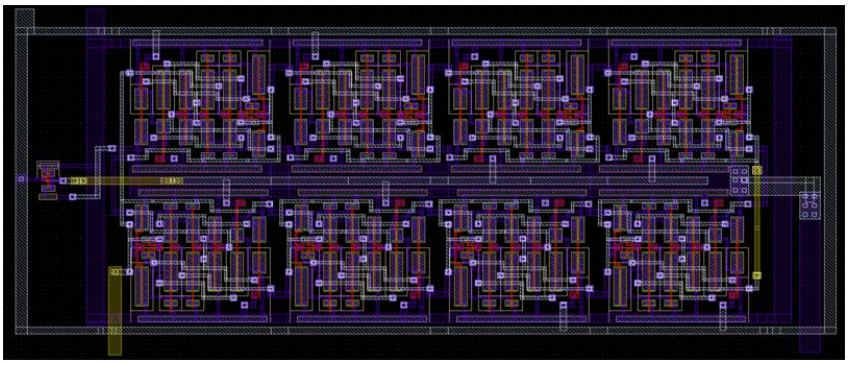


Figure 4.19 – Vue *layout* d'un anneau auto-séquencé de 8 étages

En pratique, la principale difficulté que nous avons rencontré pour la réalisation de ce circuit de test concerne l'interface de sortie. Malheureusement, les *pads* de sortie fournis par le *design kit* ont une bande passante limitée à 100 Mhz. Etant donné les fréquences que nous avons relevées en simulation (autour de 600 Mhz), nous avons été obligés de diviser les fréquences par 8 avant de sortir le signal via le plot de sortie. Cette limitation en fréquence justifie également le choix de la cellule de Muller conventionnelle qui permet d'implanter des anneaux avec des fréquences relativement basses.

5.2. Simulations électriques

Les simulations électriques ont permis de mettre en évidence deux modes d'oscillation (régulier et en rafale) en fonction du taux d'occupation des anneaux. Pour un anneau à 32 étages, le mode régulier est obtenu entre 14 et 20 jetons. Nous avons relevé les paramètres statiques des cellules : $D_{ff} = 530$ ps, $D_{rr} = 470$ ps ($D_{ff}/D_{rr} = 1.13$) et $D_{charlie} = 130$ ps. Fig. 4.20 montre la répartition des phases d'un anneau auto-séquencé dont le nombre d'événements (4) est premier avec le nombre d'étages (7). Après un temps de démarrage durant quelques périodes d'oscillation, les événements se répartissent uniformément le long d'une demi-période d'oscillation de l'anneau. Ici la résolution de phase relevée est d'environ 90 ps.

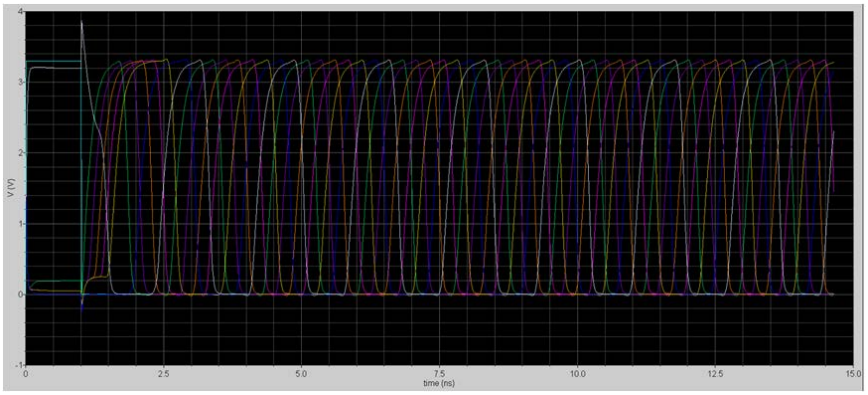


Figure 4.20 – Répartition des phases d'un anneau auto-séquencé à 7 étages et 4 jetons

Nous avons relevé dans Fig. 4.21 les fréquences d'un anneau auto-séquencé à 32 étages en fonction de son taux d'occupation. On retrouve une courbe sous forme de cloche avec une fréquence maximale autour de 16 jetons. On remarque aussi la légère dissymétrie de la courbe due au rapport $D_{ff}/D_{rr} = 1.13$. Enfin, nous avons effectué une simulation post-routage d'un anneau auto-séquencé de la configuration avec 16 jetons, nous avons observé une chute de fréquence de 74 Mhz (due aux interconnexions et aux capacités parasites).

5.3. Mesures

Le circuit réalisé contient 8 configurations d'anneaux auto-séquencés, notées STR<L>_<N>J, où $< L >$ est le nombre d'étages et $< N >$ est le nombre de jetons. Les configurations implantées sont les suivantes : STR4_2J, STR8_4J, STR16_8J, STR32_8J, STR32_12J, STR32_16J, STR32_20J et STR63_32J. Nous mesurons leur signal de sortie divisé par 8 à l'oscilloscope. Cependant, cette mesure n'est pas très précise à cause de plusieurs facteurs. D'abord, les plots de sortie ont une bande passante limitée théoriquement à 100 MHz mais il semble que les signaux soient affectés à partir d'une cinquantaine de MHz (cela se voit au niveau des pentes). Par ailleurs le standard de transmission utilisé n'est pas différentiel (la sonde utilisée est une sonde linéaire

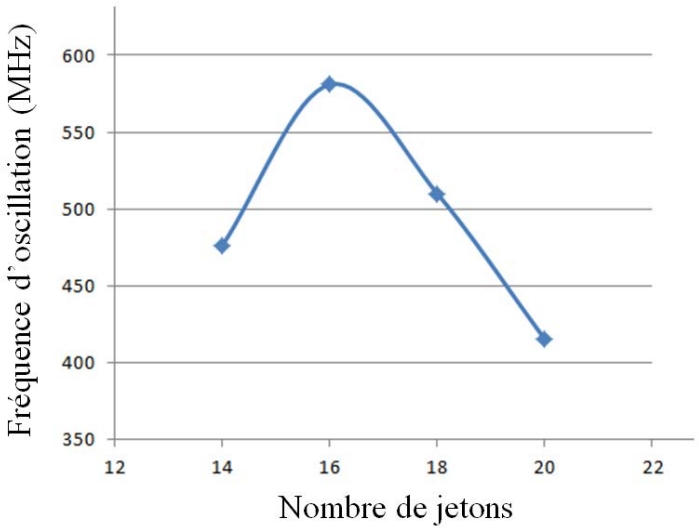


Figure 4.21 – Fréquence d'oscillation d'un anneau auto-séquencé à 32 étages en fonction de son taux d'occupation

classique). Enfin, si le calcul de la fréquence moyenne est évident à partir du signal divisé, ce n'est pas le cas pour l'écart-type de la période (amplitude du *jitter*).

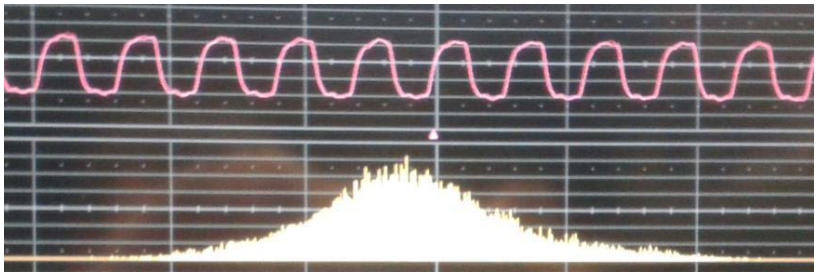


Figure 4.22 – Signal de sortie d'un anneau divisé par 8 et distribution des périodes divisées

Fig. 4.22 représente le signal de sortie divisé par 8 de la configuration STR32_20J et la distribution de ses périodes. Il est assez simple d'exploiter la mesure sur le signal divisé pour calculer la fréquence moyenne des anneaux (il suffit de multiplier la fréquence relevée par 8). Cependant, le calcul est plus complexe en ce qui concerne l'écart-type. Les mesures qui font intervenir des durées longues ont tendance à être très influencées par les phénomènes de *flicker*. Dans un cas idéal, si les réalisations de k périodes successives sont issues de lois normales indépendantes de moyenne T_{moy} et d'écart-type σ, alors leur somme peut être modélisée par une loi normale de moyenne kT_{moy} et d'écart-type $\sqrt{k}\sigma$ (les variances, *i.e* les carrés des écarts-type, s'ajoutent quand les variables aléatoires sont indépendantes). On parle d'accumulation en racine (k^n avec $n = 1/2$). Cependant l'expérience tend à démontrer que les lois d'accumulation en racine ne fonctionnent que pour de faibles valeurs de k. Dans le cas général, la courbe suit une loi en k^n avec n qui grandit plus k

est grand. Dans notre cas ($k = 8$), nous proposons d'estimer le *jitter* des anneaux implantés en utilisant une loi en k^n avec $n = 1$ et $n = 1/2$, sachant que la valeur réelle est très probablement comprise entre ces deux estimations.

Tab. 4.5 résume les mesures sur les 8 configurations d'anneaux auto-séquencés implantées. Toutes les configurations présentent un mode d'oscillation régulier sauf la configuration STR32_8J, ce qui était attendu au vu des résultats de simulation. Il est étonnant de constater la chute de fréquence, pas très importante, mais qui semble effectivement présente (comme dans les implantations FPGA) pour des configurations qui présentent le même taux d'occupation. Ceci ne semble pas lié au routage des anneaux car les *layouts* sont sensiblement similaires (les étages viennent se rajouter au milieu du *layout* de Fig. 4.19). Mais là encore, notre analyse basée sur le comportement temporel des anneaux ne prend pas en compte des aspects électriques comme la puissance du signal (celle-ci augmente, ainsi que la consommation de l'anneau, quand on augmente le nombre d'étages et d'événements qui circulent). On retrouve néanmoins le profil en cloche pour la courbe de fréquence d'un anneau dont le nombre d'étages est fixé (ici 32).

L	N_T	mode	Fréquence	σ_p (loi en racine)	σ_p (loi linéaire)
4	2	régulier	592 MHz	11.5 ps	4.1 ps
8	4	régulier	581 MHz	11.8 ps	4.2 ps
16	8	régulier	541 MHz	12.1 ps	4.3 ps
32	8	rafale	-	-	-
32	12	régulier	275 MHz	12.2 ps	4.3 ps
32	16	régulier	469 MHz	10.8 ps	3.8 ps
32	20	régulier	405 MHz	10.5 ps	3.7 ps
63	32	régulier	410 MHz	11.6 ps	4.1 ps

Table 4.5 – Mesures de fréquence et de *jitter* pour des anneaux auto-séquencés dans une cible CMOS 350 nm

En ce qui concerne le *jitter*, l'estimation avec une loi linéaire donne des valeurs autour de 4 ps. Comme pour l'implantation FPGA, il semble que la précision de la mesure ne permette pas de discriminer la variation du *jitter* en fonction du taux d'occupation de l'anneau. Cependant, il apparait que ces valeurs sont globalement très faibles et constantes (en particulier elles n'augmentent pas avec le nombre d'étages).

5.4. Synthèse des résultats

L'implantation d'anneaux auto-séquencés dans les cibles ASIC est assez simple dans l'optique où le concepteur peut contrôler tous les paramètres de *design* (conception de type *full-custom*). Les risques d'une mauvaise conception (par exemple la présence de *bottleneck*) peuvent être facilement anticipés au vu de l'efficacité des simulateurs électriques.

Comme pour les implantations FPGA, le *jitter* dans les anneaux auto-séquencés est globalement faible et ne varie ni avec le taux d'occupation des anneaux ni avec leur nombre d'étages (du moins au vu de la résolution de nos mesures). Encore une fois, cela suggère que le *jitter* mesuré dans le signal de sortie est majoritairement dû au bruit local de l'étage où on effectue la mesure.

Ceci est très important dans le contexte du STRNG comme on le verra dans le chapitre suivant. Par ailleurs, les mesures de fréquences d'oscillation et d'amplitude de *jitter* seront exploitées dans la suite pour choisir le nombre d'étages du STRNG afin de garantir un taux d'entropie souhaité en sortie.

6. Conclusion

Dans la première partie de ce chapitre, nous avons modélisé le bruit dans des anneaux à inverseurs et dans des anneaux auto-séquencés. Le résultat important de cette partie concerne la manière dont les variations temporelles se propagent dans ces anneaux. Dans les anneaux à inverseurs, le bruit généré dans un étage peut être mesuré également dans tous les étages de l'anneau. Le *jitter* sur la période d'oscillation est donc issu de l'accumulation du bruit généré séparément dans chaque étage. *A contrario*, dans un anneau auto-séquencé, le bruit généré dans un étage est de moins en moins mesurable plus on s'éloigne de cet étage. Le *jitter* sur la période d'oscillation est donc principalement issu du bruit généré dans l'étage où on effectue la mesure. Cet effet est d'autant plus marqué que le taux d'occupation est optimal ($\frac{D_{ff}}{D_{rr}} = \frac{N_T}{N_B}$) et l'amplitude de l'effet *Charlie* est grande. Dans le cas optimal, l'écart-type sur la période d'oscillation (σ_{period}) vaut environ 1.7 fois l'écart-type du délai de propagation des étages (σ_g).

Ensuite, nous nous sommes intéressé à des implantations d'anneaux auto-séquencés dans des cibles FPGA (Altera Cyclone III et Xilinx Virtex 5) et ASIC (technologie CMOS 350 nm). Toutes les configurations testées ont présenté un profil de *jitter* gaussien, dont l'amplitude est très faible et ne varie pas significativement ni avec le nombre d'étages ni avec le taux d'occupation des anneaux. Selon nos mesures, nous avons constaté des valeurs de σ_{period} entre $1.5\sigma_g$ et $2\sigma_g$ (étant donné les faibles valeurs, on est à la limite des outils de mesure). Enfin, il est important de préciser que les anneaux auto-séquencés, quelque soit leur taille, fonctionnement à des fréquences généralement hautes (semblables à celles générées par des anneaux à inverseurs de moins de 5 étages). Ceci permet de limiter l'incidence de bruits dépendants de la fréquence (bruit de *flicker*, bruit en $1/F^2$ ...).

Pour conclure, nous constatons que les anneaux auto-séquencés sont de très bons candidats pour fournir une source d'entropie au sein d'un TRNG. Leur principal atout vient des effets analogiques propres à la structure de leurs étages, en particulier l'effet *Charlie*. Celui-ci régule les temps des événements au sein de l'anneau, ainsi le bruit généré dans un étage influe peu sur les temps de basculement des autres étages. De cette manière, le *jitter* relevé sur le signal de sortie est principalement dû au bruit local de cet étage. Sa faible valeur n'est pas problématique étant donné le concept du STRNG, elle est même un indicateur de qualité car on a constaté des valeurs constantes malgré des différences de fréquences de près de 200 MHz, ce qui suggère que le bruit généré a un spectre plat en fréquence. D'autre part, cette stabilité de la fréquence accrue due à la présence de l'effet *Charlie* se traduit aussi par une meilleure robustesse aux variations de tension : la source d'aléa est globalement moins influencée par des paramètres externes au circuit.

Chapitre 5.

Modélisation et sécurisation du générateur

Sommaire

1. Introduction

La sécurité d'un TRNG est liée à deux facteurs : l'imprévisibilité de sa sortie (l'entropie) et les moyens mis en oeuvre pour la garantir quelque soient les conditions (variations environnementales, attaques, etc). Pour établir l'imprévisibilité de la sortie d'un TRNG, il faut : 1) Caractériser le phénomène aléatoire exploité, le mesurer et le modéliser par une loi aléatoire adéquate. 2) Modéliser le comportement de l'extracteur afin de mettre en lien la loi caractérisant le phénomène aléatoire et les nombres en sortie du générateur (eux même représentés par des variables aléatoires). 3) Enfin, en étudiant les variables aléatoires représentant les nombres de sortie, il faut calculer l'entropie par bit de sortie, ou du moins en estimer une limite basse. Dans le contexte de cette étude, cette dernière est interprétée comme une mesure de l'imprévisibilité de la sortie : elle vaut 0 si les bits sont déterministes, et elle vaut 1 si les bits sont complètement imprévisibles, indépendants, uniformément répartis et non biaisés. Une entropie inférieure 1 signifie qu'il est possible de deviner les bits de sortie avec une probabilité non nulle (plus cette probabilité est grande plus l'entropie est faible), les séquences sont générées alors naturellement biaisées.

Dans le cas du STRNG, nous montrons dans ce chapitre que l'entropie en sortie est directement liée au réglage de la résolution de phase de l'anneau auto-séquencé en relation avec l'amplitude du *jitter*. Dans le chapitre 2, nous avons établi le lien entre la résolution de phase de l'anneau et sa configuration (nombre d'étages, nombre de jetons et période d'oscillation). Ensuite, dans le chapitre 4, nous avons caractérisé et mesuré le *jitter* dans les anneaux auto-séquencés. Nous avons constaté qu'il peut être modélisé par une loi normale dont l'écart-type représente l'amplitude du *jitter*.

Dans la première partie de ce chapitre, nous quantifions un seuil bas d'entropie par bit de sortie du STRNG en fonction des paramètres de l'anneau auto-séquencé (nombre d'étages et fréquence d'oscillation) et de l'amplitude du *jitter*. Ensuite, dans la seconde partie de ce chapitre, nous proposons des moyens pour *monitorer* l'entropie mais aussi pour protéger le générateur contre d'éventuelles attaques. Pour cela, nous proposons et analysons un modèle de menace du générateur. En se basant sur cette analyse, nous posons les bases théoriques pour augmenter la sécurité du STRNG en contrôlant son fonctionnement en temps réel ou à la demande. Ces mesures comprennent des tests spécifiques aux principe du générateur ainsi que des moyens pour *monitorer* l'entropie à sa sortie.

2. Modélisation du générateur

Chaque bit de sortie b_n du générateur, obtenu lors d'un cycle d'horloge n, peut être modélisé avec une variable aléatoire notée B_n dont les réalisations dépendent de celles du *jitter* de l'anneau auto-séquencé. L'objectif de cette section est, dans un premier temps, d'estimer un seuil bas pour $H(B_n)$, $H(B_n)$ désignant l'entropie de Shannon de la variable B_n. Ensuite, nous étudions les corrélations éventuelles entre bits successifs du STRNG afin de dériver une condition sur la fréquence d'échantillonnage pour limiter ces dépendances.

2.1. Observations et remarques préliminaires

Le modèle présenté dans cette section se base sur les observations suivantes (celles-ci proviennent des concepts et résultats présentés dans les chapitres 2, 3 et 4) :

Présence d'un jitter Gaussien issu de bruit blanc – Le modèle assume la présence d'une composante de *jitter* aléatoire ayant une distribution gaussienne à la sortie de chaque étage de l'anneau auto-séquencé. Cette composante aléatoire est en pratique toujours présente dans les circuits du fait de l'inévitable bruit thermique (un bruit blanc ayant un spectre plat), par contre d'autres composantes déterministes peuvent s'y ajouter selon les conditions de fonctionnement. Généralement, la principale difficulté consiste à mesurer l'amplitude de la composante aléatoire indépendamment d'autres composantes de bruit. Néanmoins, nous avons constaté dans le chapitre 4 que le *jitter* dans les anneaux asynchrones présente un profil de type gaussien dont l'écart-type ne varie pas significativement ni avec la configuration ni avec le nombre d'étages de l'anneau. Nous avons aussi remarqué que cet amplitude ne variait pas pour les gammes de fréquences observées (de 300 MHz à 500 MHz), ce qui suggère que ce *jitter* est décorrélé de la fréquence (le *flicker noise* est limité car les fréquences considérées sont relativement grandes). Enfin, un événement qui se propage dans l'anneau ne transporte pas indéfiniment l'information liée à une réalisation du *jitter* donnée car les temps des transitions sont auto-régulées par les effets *Charlie* et de *drafting*. Ceci est un facteur très important à prendre en compte pour l'analyse d'éventuelles dépendances entres bits successifs de l'anneau.

Jitter de l'horloge d'échantillonnage – Nous souhaitons estimer un seuil bas d'entropie associé à l'imprévisibilité du *jitter* de l'anneau auto-séquencé, et non du *jitter* de l'horloge d'échantillonnage dont on suppose que les caractéristiques ne sont pas forcément connues (elle dépendent du cadre d'utilisation du générateur). Elle peut provenir d'un anneau à inverseurs, d'une PLL, d'un quartz, etc. Elle peut être sujette à un *jitter* de type aléatoire ou déterministe d'amplitude forte ou faible, et peut avoir une fréquence quelconque. Les taux d'entropie calculés dans la suite reflètent donc uniquement l'imprévisibilité du *jitter* de l'anneau auto-séquencé, les taux d'entropie réels sont nécessairement supérieurs si on prend en compte l'imprévisibilité des instants d'échantillonnage.

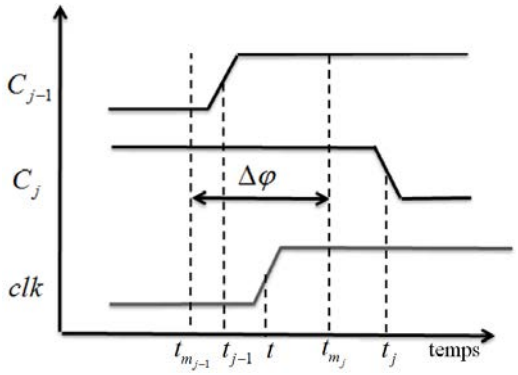


Figure 5.1 – Représentation de deux événements successifs dans le temps dans un anneau auto-séquencé (C_j et C_{j-1} ne sont pas des étages adjacents)

Signaux de sortie de l'anneau – Les signaux de sortie de l'anneau auto-séquencé $(C_i)_{1\leq i\leq L}$ produisent L transitions électriques sujettes au *jitter*, dont les temps moyens (notés $(tm_i)_{1\leq i\leq L}$) sont uniformément distribués sur une demi-période d'oscillation de l'anneau. Ces signaux de sortie sont ensuite ré-indexés selon ces temps moyens de manière à ce que $tm_1 \leq tm_2 \leq ... \leq tm_L$. On dénote $\triangle\varphi$ le temps moyen entre deux événements successifs dans l'anneau : $tm_{i+1} - tm_i = \triangle\varphi$. $\triangle\varphi$ correspond à la résolution de phase de l'anneau, et peut être calculé à l'aide de Eq. (2.8). Fig. 5.1 représente deux événements successifs dans le temps dans un anneau auto-séquencé (j et $j-1$ correspondent aux nouveaux index). Le lien entre les nouveaux indexes des étages et leur ordre réel dans la structure dépend en pratique du taux d'occupation de l'anneau. Dans le cas où $\frac{N_T}{N_B} \simeq \frac{D_{ff}}{D_{rr}} \simeq 1$, ce lien est très simple, les étages sont ré-indexés selon l'ordre suivant : 0, 2, 4, 6, ..., L-1 (si L est impair), 1, 3, 5, ..., L-2. Dans ce cas la résolution de phase vaut le temps de synchronisation s (ou temps de séparation) entre les deux entrées d'un étage dans le régime permanent. Plus s est petit (*i.e.* plus la résolution de phase est fine), plus l'influence de l'effet *Charlie* est importante (et donc plus l'effet d'auto-régulation temporelle de l'anneau est important). Le temps effectif d'un événement, noté t_j est quand à lui fonction des réalisations du *jitter*. Il peut être décrit par une loi normale dont la moyenne est t_{m_j} et donc l'écart-type σ représente l'amplitude du *jitter* généré localement dans un étage de l'anneau. Enfin, chaque signal C_i est échantillonné au même instant t, les signaux en sortie des bascules, notés $(s_i)_{1\leq i\leq L}$ sont ensuite combinés à l'aide d'une fonction XOR. ψ est le signal résultant.

2.2. Modélisation de l'extraction d'entropie

Chaque bit b_n en sortie du signal ψ au cycle d'échantillonnage n est une variable aléatoire notée B_n. Nous souhaitons estimer $H(B_n)$, il faut donc calculer la probabilité que la variable B_n prenne la valeur '0' ou '1'. Ces probabilités dépendent de l'instant relatif de l'échantillonnage par rapport aux fronts de l'anneau auto-séquencé, mais aussi de la réalisation du *jitter*.

La variable B_n est associée à un instant d'échantillonnage t (correspondant au $n^{ième}$ cycle d'échantillonnage). Comme les phases de l'anneau sont uniformément réparties et équi-distantes de $\triangle\varphi$, alors il existe deux signaux C_{j-1} and C_j tels que $|t - tm_j| \leq \frac{\triangle\varphi}{2}$ ou $|t - tm_{j-1}| \leq \frac{\triangle\varphi}{2}$ et $tm_{j-1} \leq t \leq tm_j$. Nous positionnons l'origine des temps au milieu de l'intervalle moyen entre les deux événements (de manière à ce que $tm_j + tm_{j-1} = 0$) comme cela est illustré dans Fig. 5.2.

L'instant effectif de basculement du signal issu de C_j est une variable aléatoire X_j décrite par une loi normale dont la moyenne est $\frac{\triangle\varphi}{2}$ et dont la variance est σ^2 (la variance correspond au carré de l'écart-type). Elle est notée : $X_j = \mathcal{N}(\frac{\triangle\varphi}{2}, \sigma^2)$. L'instant effectif de basculement du signal issu de C_j est une variable aléatoire X_{j-1} décrite par une loi normale dont la moyenne est $-\frac{\triangle\varphi}{2}$ et dont la variance est σ^2. Elle est notée : $X_{j-1} = \mathcal{N}(-\frac{\triangle\varphi}{2}, \sigma^2)$.

Il est très important de préciser que les variables X_j et X_{j-1} sont rigoureusement indépendantes par construction : elles proviennent de deux étages où deux événements se sont produit en un temps très court (avec $\triangle\varphi$ d'écart). La transition sur l'étage C_j n'est pas générée à partir de la transition sur l'étage C_{j-1} comme cela serait le cas dans un anneau à inverseurs. D'ailleurs, il n'est absolument exclu que pour une réalisation particulière du *jitter*, le front du signal C_j arrive avant celui du signal

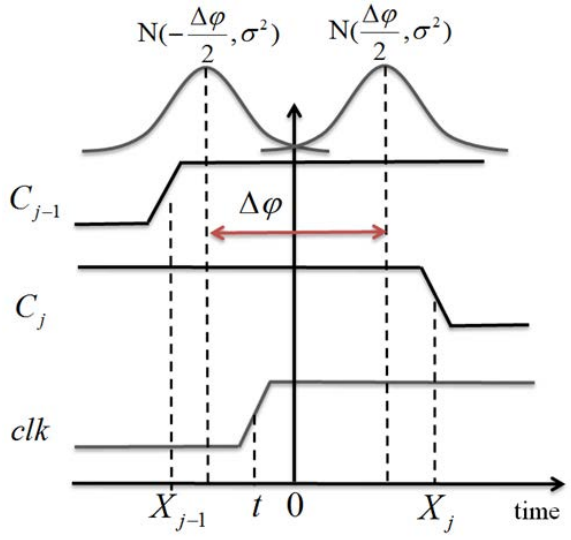


Figure 5.2 – Modélisation de l'extraction d'entropie

C_{j-1}.

Le signal ψ (sortie de l'arbe de XOR) peut-être décomposé en la somme $\psi = \omega + \mu$ avec :

$$\omega = s_j \oplus s_{j-1} \quad , \quad \mu = \oplus(s_i)_{i \neq j, i \neq j-1} \tag{5.1}$$

Notons s_{k_n} le bit échantillonné à la sortie de la bascule k au cycle d'échantillonnage n (i.e. la valeur de s_k au cycle n). Notons ω_n la valeur du signal ω au cycle n et μ_n la valeur du signal μ au cycle n. Comme les $s_{i_n 1 \leq i \leq L}$ sont des échantillons rigoureusement indépendants, alors $H(B_n) \geq H(\omega_n)$. On peut donc déduire un seuil bas d'entropie par bit de sortie du générateur en estimant l'entropie des sorties des deux bascules dont l'échantillonnage s'effectue au plus près des fronts théoriques de l'anneau auto-séquencé (i.e. les sorties s_{j-1} et s_j).

En pratique, les signaux $(s_i)_{i \neq j, i \neq j-1}$ ne contribuent à l'entropie de la somme que lorsque l'amplitude du *jitter* est très grande par rapport à $\triangle\varphi$, de manière à ce que la gaussienne représentée dans Fig. 5.2 recouvre plusieurs phases de l'anneau. Or dans ce cas ($\sigma \gg \triangle\varphi$), $H(\omega_n) \simeq 1$ et donc $H(B_n) \simeq 1$: il n'est donc plus nécessaire de calculer l'entropie des autres bits. De manière générale, quand les bits $(s_i)_{i \neq j, i \neq j-1}$ commencent à avoir une entropie non négligeable, alors l'entropie du bit échantillonné dans ω s'approche déjà de 1 (le calcul qui permet de vérifier cela est le même que celui présenté dans les deux prochaines sections, précisément Eq. (5.7) et Eq. (5.8)).

Dans la suite nous nous intéressons donc uniquement à l'entropie du signal ω (composé de la somme de C_j et C_{j-1} de Fig. 5.2) en supposant que les autres bascules produisent une valeur déterministe. Ceci revient à considérer que σ n'est pas très grand par rapport à $\triangle\varphi$ (autrement il n'est pas nécessaire de calculer l'entropie car elle est systématiquement proche de 1). Notons $'u'$

('u' vaut '1' ou '0') la valeur échantillonnée dans le signal μ au cycle d'échantillonnage n, on a :

$$\mu_n =' u' \quad , \quad H(\mu_n) \simeq 0 \tag{5.2}$$

Autrement dit on, suppose que le signal μ vaut 'u' avec une probabilité très grande. La valeur 'u' dépend uniquement de l'instant relatif d'échantillonnage par rapport au front de l'anneau (et non des réalisations de *jitter* de l'anneau), elle varie selon la déviation de phase de l'horloge d'échantillonnage si celle-ci est externe, représentant la partie pseudo-déterministe du générateur. Elle est constante si l'échantillonnage est interne. Finalement, on peut synthétiser l'ensemble de ces remarques dans les équations suivantes :

$$H(B_n) \simeq H(\omega_n) \quad , \quad \omega_n = s_{j_n} \oplus s_{j-1_n}, \tag{5.3}$$

où s_{j_n} et s_{j-1_n} dépendent des réalisations des variables aléatoires X_j and X_{j-1}, décrites par les lois de probabilités suivantes :

$$X_j = \mathcal{N}(\frac{\triangle\varphi}{2}, \sigma^2) \;\; and \;\; X_{j-1} = \mathcal{N}(-\frac{\triangle\varphi}{2}, \sigma^2) \tag{5.4}$$

2.3. Calcul des probabilités

Pour un instant d'échantillonnage t fixé (correspondant à un cycle d'échantillonnage n), la valeur échantillonnée dans le signal ω_n dépend de la position relative des fronts des étages C_j et C_{j-1}, qui dépend des réalisations des variables aléatoires X_j et X_{j-1} comme cela est représenté dans Fig. 5.2. Tab. 5.1 donne la valeur de ω_n et de la variable B_n en fonction des réalisations des variables aléatoires X_j et X_{j-1} (nous rappelons que 'u' représente la partie déterministe des échantillons sommés).

$X_{j-1} \leq t$	$X_j \leq t$	ω_n	B_n
vrai	faux	'1'	$\bar{u}$
faux	vrai	'0'	u
vrai	faux	'0'	u
vrai	vrai	'1'	$\bar{u}$

Table 5.1 – Valeurs de ω_n et B_n en fonction des réalisations de X_{j-1} et X_j pour un instant d'échantillonnage t fixé

Nous souhaitons calculer la probabilité que le bit B_n prenne la valeur 'u', que nous notons dans la suite $P(u)$. Soit $p = P(X_j \leq t)$ la probabilité que $X_j \leq t$, et $p' = P(X_{j-1} \leq t)$ la probabilité que $X_{j-1} \leq t$. Comme ces deux événements sont rigoureusement indépendants, et selon Tab. 5.1, $P(u)$ peut être exprimée ainsi :

$$P(u) = p + p' - 2pp' \tag{5.5}$$

La variable aléatoire X_j étant définie par une loi normale (Eq. (5.4)), on peut calculer la probabilité $p = P(X_j \leq t)$ en utilisant la fonction de répartition de la loi normale centrée réduite $\mathcal{N}(0, 1)$ via un changement de variable. Celle-ci est définie ainsi :

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-t^2}{2}} dt \quad , \; x \in \mathbb{R}$$

La fonction de répartition de la loi normale centrée réduite décrit la probabilité que la réalisation de la variable aléatoire associée soit inférieure à la valeur x. D'après Eq. (5.4) et Eq. (2.3.), on exprime p et p' en fonction de σ, $\triangle\varphi$ et t :

$$p = \Phi(\frac{t - \frac{\triangle\varphi}{2}}{\sigma}) \quad and \quad p' = \Phi(\frac{t + \frac{\triangle\varphi}{2}}{\sigma}) \tag{5.6}$$

Finalement, on peut exprimer la probabilité $P(u)$ en fonction de l'instant d'échantillonnage t, de la résolution de phase moyenne $\triangle\varphi$ et de l'amplitude du *jitter* σ :

$$P(u) = \Phi(\frac{t - \frac{\triangle\varphi}{2}}{\sigma}) + \Phi(\frac{t + \frac{\triangle\varphi}{2}}{\sigma}) - 2\Phi(\frac{t - \frac{\triangle\varphi}{2}}{\sigma})\Phi(\frac{t + \frac{\triangle\varphi}{2}}{\sigma}) \tag{5.7}$$

2.4. Estimateurs de biais et d'entropie

Comme la variable B_n peut prendre deux valeurs u ou $\bar{u}$, et sachant que $P(\bar{u}) = 1 - P(u)$ (les deux événements s'excluent), alors l'entropie de Shannon de la variable B_n, notée H dans la suite, est définie par la formule suivante :

$$H = H(B_n) = -P(u)log_2(P(u)) - (1 - P(u))log_2(1 - P(u)) \tag{5.8}$$

Nous définissons le biais de la variable B_n, noté B, par la formule suivante :

$$B = \mathcal{B}(B_n) = \frac{1}{2} - P(u) \tag{5.9}$$

$P(u)$ est calculé grâce à Eq. (5.7). Ainsi l'entropie du bit de sortie du générateur H est fonction de t, σ et de $\triangle\varphi$. Fig.5.3 représente le biais absolu $|B|$ et l'entropie H de la variable B_n en fonction de l'instant d'échantillonnage pour différentes valeurs d'amplitude du *jitter* (dans l'exemple de la courbe $\triangle\varphi$ vaut 10 unités de temps). L'entropie est maximale lorsque l'échantillonnage s'effectue au même moment que l'instant moyen des fronts ($t = \frac{\triangle\varphi}{2}$ et $t = \frac{-\triangle\varphi}{2}$). A l'inverse, elle est minimale quand l'échantillonnage s'effectue le plus loin des instants moyens des fronts ($t = 0$). D'autre part, plus l'amplitude du *jitter* σ est grande, et plus la résolution de phase de l'anneau $\triangle\varphi$ est fine, plus la valeur minimale de H (*i.e* quand $t = 0$) est grande. En particulier, on voit que pour la courbe $\triangle\varphi = \sigma/4$, ce seuil minimal est très proche de 1. Comme $\triangle\varphi$ peut être ajusté en choisissant judicieusement le nombre d'étages de l'anneau, alors il est toujours possible de régler ce seuil aussi haut que voulu quelque soit l'amplitude du *jitter*.

Un seuil haut de biais (noté $|B|_m$) et un seuil bas d'entropie (noté H_m) de la variable B_n peuvent être obtenus en remplaçant l'expression générale de $P(u)$ par $P(u)_{t=0}$ dans Eq. (5.8) et Eq. (5.9). D'après Eq. (5.7), et sachant que $\Phi(-x) = 1 - \Phi(x)$ ($x \in \mathbb{R}$), alors $P(u)_{t=0}$ peut être exprimée ainsi :

$$P(u)_{t=0} = 1 - 2\Phi(\frac{\triangle\varphi}{2\sigma}) + 2(\Phi(\frac{\triangle\varphi}{2\sigma}))^2 \tag{5.10}$$

La résolution de phase moyenne $\triangle\varphi$ est exprimée en fonction du nombre d'étages de l'anneau

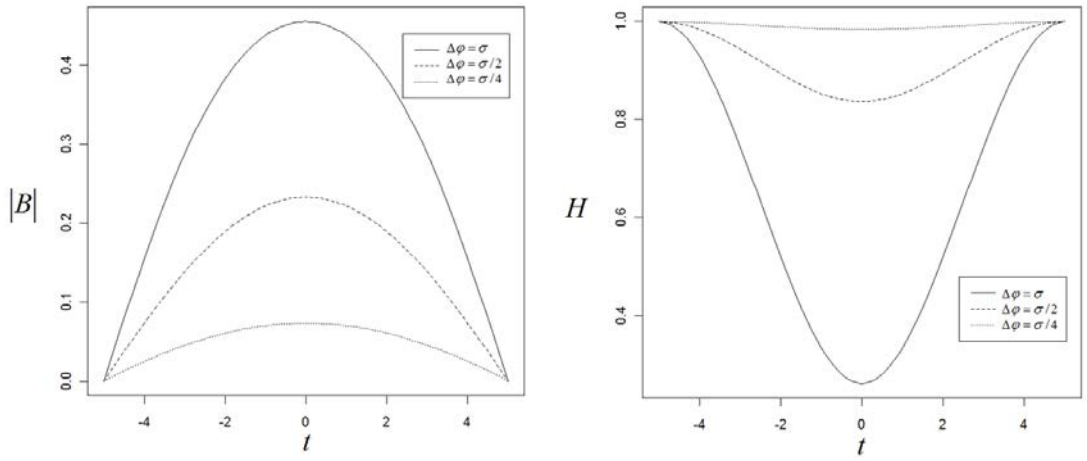


Figure 5.3 – Biais absolu $|B|$ et entropie H du bit B_n en fonction de l'instant d'échantillonnage t, pour différentes valeurs d'amplitude de *jitter* σ relatives à la résolution de phase $\triangle\varphi$

et de sa période d'oscillation par la relation représentée dans Eq. (2.8) ($\triangle\varphi = T/2L$). Finalement, l'expression du seuil bas d'entropie du bit de sortie B_n est la suivante :

$$H_m = -P(u)_{t=0} log_2(P(u)_{t=0}) - (1 - P(u)_{t=0}) log_2(1 - P(u)_{t=0}) \tag{5.11}$$

avec :

$$P(u)_{t=0} = 1 - 2\Phi(\frac{T}{4L\sigma}) + 2(\Phi(\frac{T}{4L\sigma}))^2 \tag{5.12}$$

H_m est indépendant de l'instant d'échantillonnage, il est fonction de l'amplitude du *jitter* σ, du nombre d'étages de l'anneau L et de sa période d'oscillation T. Nous avons tracé $|B|_m$ (seuil haut du biais) et H_m (seuil bas de l'entropie) en fonction de L (nombre d'étages) pour différentes valeurs de σ/T dans Fig. 5.4. Nous supposons que la période d'oscillation T est maintenue constante en augmentant le nombre d'étages et en choisissant judicieusement le nombre d'événements tout en respectant la condition qu'il soit premier avec le nombre d'étages. On remarque que H_m augmente de manière non linéaire avec le nombre d'étages : elle augmente de manière quasi-exponentielle pour de faibles valeurs de L, par contre elle augmente très lentement pour les grandes valeurs. Par exemple, pour la courbe $\sigma/T = 0.001$, il faut environ 300 étages pour obtenir $H_m > 0.90$, mais il faut plus de 600 étages pour que $H_m > 0.99$.

En conclusion, le réglage de l'entropie en sortie du STRNG se fait très simplement en choisissant le nombre d'étages de l'anneau auto-séquencé en fonction de l'amplitude mesurée du *jitter* et de la période mesurée de l'anneau auto-séquencé. Ceci permet d'adapter l'architecture en prenant en compte la cible d'implantation et l'évolution des technologies.

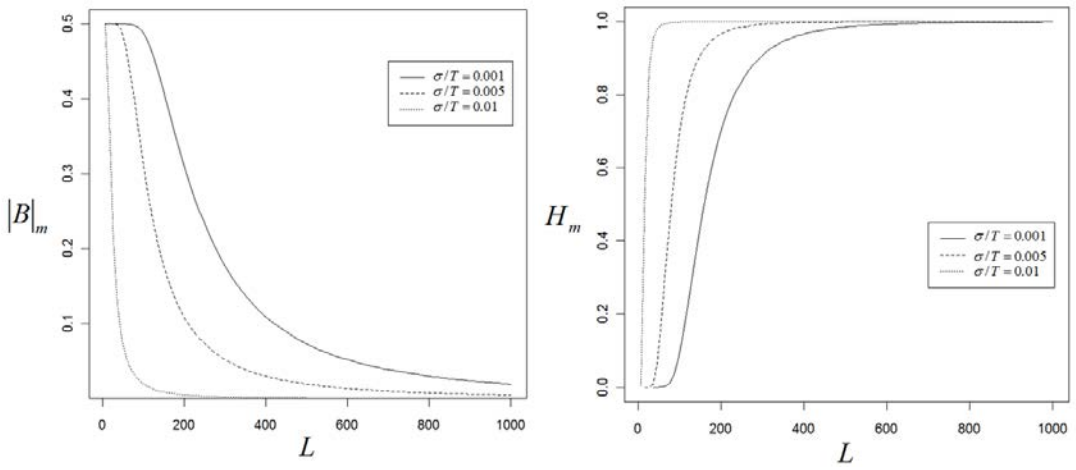


Figure 5.4 – Seuil haut du biais absolu $|B|_m$ et seuil bas d'entropie H_m du bit B_n en fonction du nombre d'étages de l'anneau auto-séquencé L, pour différentes valeurs d'amplitude de *jitter* σ relatives à la période d'oscillation T de l'anneau

2.5. Etude des corrélations entre bits successifs

Dans la section précédente, nous avons estimé $H(B_n)$ l'entropie du bit B_n. Dans ce calcul, nous avons supposé que la phase moyenne entre deux signaux au cycle d'échantillonnage n est entièrement déterminée par les paramètres statiques de l'anneau, et qu'elle est indépendante des réalisations de *jitter* aux cycles d'échantillonnage précédents : ceci revient à supposer que l'anneau régule systématiquement et immédiatement les variations temporelles dues au bruit (en négligeant le régime transitoire même si celui-ci est très éphémère au vu des faibles variations introduites par une réalisation de *jitter* donnée). Dans la réalité, on peut imaginer qu'une information (une variation temporelle due à une réalisation du *jitter*) a une durée de vie limitée dans l'anneau, comme le montrent les résultats du chapitre 4 : elle se perd au fur et à mesure que l'événement qui la transporte se propage dans la structure. Dans cette section, nous souhaitons analyser de près ce phénomène afin de dériver une condition sur le pas d'échantillonnage permettant d'obtenir des bits de sortie indépendants.

Fig. 5.5 illustre la dépendance immédiate entre les positions des fronts dans un anneau auto-séquencé où $D_{ff} = D_{rr}$ et $N_T = \frac{L+1}{2}$ (les indexes des étages de cette figure correspondent à leur ordre original dans la structure). La résolution de phase de l'anneau correspond à la phase entre un signal C_{j-1} et un signal C_{j+1}. La position d'un événement j, notée $t_{j,m}$ dans Fig. 5.5 (l'indice m permet de discriminer les positions successives de cet événement quand il se propage dans les étages suivants), influe sur sa position quand il se déplace à l'étage suivant ($t_{j,m+1}$), mais aussi sur la position de l'événement $j-1$ lorsque lorsque celui-ci se déplace également (notée $t_{j-1,m+1}$ dans la figure) via le retour du signal d'acquittement. En moyenne, *i.e* si on ne prend pas en compte le *jitter*, le délai de propagation d'un événement dans le régime dynamique est $D_m = \frac{N_T T}{2L}$, le délai du signal de retour d'acquittement est alors $D'_m = \frac{N_T T}{2L} - \triangle\varphi_m$, où $\triangle\varphi_m$ est la résolution de phase de l'anneau.

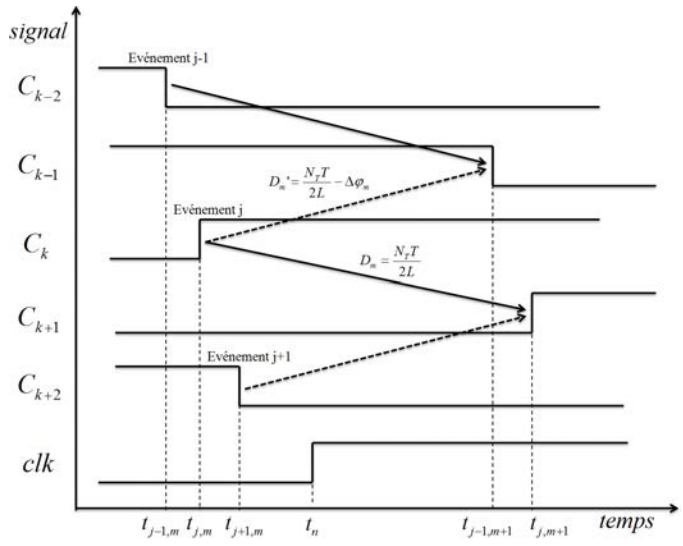


Figure 5.5 – Illustration des dépendances entre les positions des fronts dans un anneau auto-séquencé

Soit t_n l'instant d'échantillonnage au cycle n. On peut considérer qu'on a une dépendance directe entre un bit échantillonné en sortie du TRNG B_{n+1} et son prédécesseur B_n si le pas d'échantillonnage est proche de $D_m = \frac{N_T T}{2L}$ ou $D'_m = \frac{N_T T}{2L} - \triangle\varphi_m$. On a une dépendance de second ordre (plus faible), si on échantillonne en avec un pas proche de $2D_m$ ou $D_m + D'_m$ ou $2D'_m$. Puis une dépendance de troisième ordre si on échantillonne avec un pas proche de $3D_m$ ou $2D_m + D'_m$ ou $D_m + 2D'_m$ ou $3D'_m$, et ainsi de suite pour les dépendances d'ordre supérieur. Par contre si le pas d'échantillonnage n'est pas proche de ces valeurs, alors les échantillons peuvent être considérés comme indépendants. Par exemple, on peut très bien échantillonner autour d'un événement j puis autour d'un événement $j+k$ ($k \geq 2$) avec un pas inférieur à Dm' pour obtenir deux bits successifs indépendants. On peut remarquer que plus l'anneau est long, et plus il comporte d'événements (*i.e.* plus N_T est grand), plus est facile de respecter cette condition sur le pas d'échantillonnage (en effet on a moins de chance de ré-échantillonner deux fois de suite autour du même événement alors qu'il se propage entre deux étages). De plus, en pratique, nous verrons dans la suite que $\frac{N_T T}{2L}$ est un pas d'échantillonnage beaucoup trop rapide à la vue des débits permis par les technologies utilisées. Par exemple, si $N_T \simeq L/2$, ce pas d'échantillonnage correspond à une fréquence quatre fois plus rapide que celle de l'anneau auto-séquencé, il est proche du délai de propagation d'une porte logique de base de la technologie utilisée.

Dans un cas plus général, sans connaître précisément le pas d'échantillonnage, on peut dériver une condition sur la fréquence d'échantillonnage pour pouvoir raisonnablement négliger ces dépendances (même si ré-échantillonne autour du même événement, mais après qu'il se soit propagé durant un temps suffisamment long). On considère que si le pas d'échantillonnage est supérieur à $k \times \frac{N_T T}{2L}$, alors dans le pire cas, il y a une dépendance de $k^{ième}$ ordre entre B_{n+1} et B_n. Nous souhaitons évaluer (grâce à la simulation numérique) quel est l'ordre k minimal pour lequel on

peut raisonnablement négliger cette dépendance.

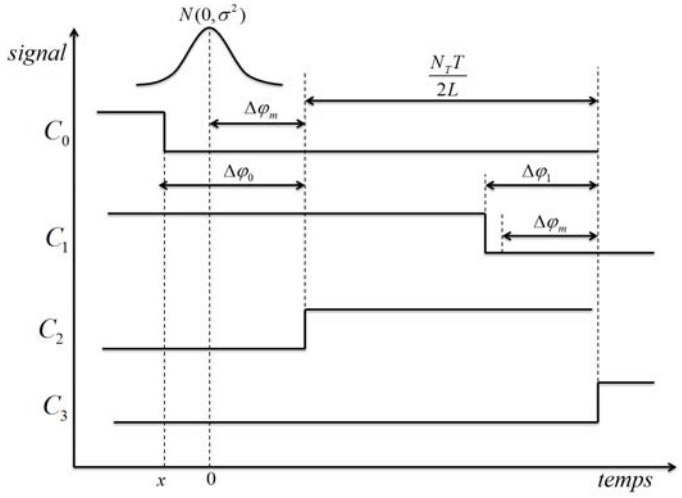


Figure 5.6 – Influence d'une réalisation du *jitter* x sur les prochaines phases autour de l'événement qui la transporte dans un anneau auto-séquencé où $D_{ff} = D_{rr}$ et $N_T = \frac{L+1}{2}$

Le cadre de la simulation numérique présentée dans cette section est celui de la section 3 du chapitre 4. L'objectif est d'analyser la durée de vie d'une réalisation du *jitter* au fur et à mesure que l'événement qui la transporte se propage dans l'anneau. Dans Fig. 5.6, l'étage C_0 est modélisé par une source de bruit, il produit des variations temporelles, alors que les autres étages sont considérés idéaux. La résolution de phase moyenne de l'anneau est notée $\triangle\varphi_m$ (elle correspond à la résolution de phase lorsque tous les étages sont idéaux). La position effective d'un front du signal C_0 est une variable aléatoire X suivant une loi normale de moyenne 0 (selon l'axe placé dans Fig. 5.6) et d'écart-type $\sigma = 10$ ps. En fonction de la réalisation x de X, associée à une phase effective $\triangle\varphi_0$, nous calculons la probabilité de cette réalisation (plus précisément la probabilité $Pr(|X| > |x|)$, qui représente la probabilité d'être dans une situation plus "défavorable"). Ensuite nous relevons la phase $\triangle\varphi_1$ lorsque l'événement présent dans C_0 se propage à l'étage suivant C_1. Puis de la même manière, nous relevons la phase $\triangle\varphi_2$ lorsque ce même événement se propage à l'étage C_2, et $\triangle\varphi_3$ quand il se propage à l'étage C_3. Rappelons que le temps de propagation dans le régime dynamique pour la configuration multiphasée vaut $\frac{N_T T}{2L}$.

Nous effectuons cette expérience pour 3 configurations d'anneau : une avec 31 étages ($\triangle\varphi_m =$ 34 ps), une avec 63 étages ($\triangle\varphi_m = 16$ ps) et une 127 ($\triangle\varphi_m = 8$ ps). Nous rappelons que plus la résolution de phase est fine, plus l'influence de l'effet *Charlie* est importante. Les paramètres de la simulation (amplitude de l'effet Charlie, délais statiques...) sont proches de ceux relevés sur le circuit CMOS 350 nm (*c.f.* la section 5.2 du chapitre 4). Par contre, l'amplitude du *jitter* ($\sigma = 10$ ps) est environ 5 fois supérieure aux valeurs réelles (pour avoir une meilleure précision du simulateur). Nous souhaitons évaluer à quel point la réalisation du *jitter* de l'étage C_0 influe sur les prochaines phases autour d'un événement se propageant de C_0 à C_1, puis quand il se propage vers C_2 ($\triangle\varphi_2$ et $\triangle\varphi_3$). Si ces phases sont égales à $\triangle\varphi_m$, alors on peut considérer qu'elles sont

indépendantes des réalisations du *jitter* de C_0.

$\triangle\varphi_m$	x	$Pr(\|X\| > \|x\|)$	$\triangle\varphi_0$	$\triangle\varphi_1$	$\triangle\varphi_2$	$\triangle\varphi_3$
34 ps	-17 ps	0.10	51 ps	40 ps	32 ps	34 ps
34 ps	-12 ps	0.22	46 ps	39 ps	32 ps	34 ps
34 ps	12 ps	0.22	22 ps	29 ps	36 ps	34 ps
34 ps	21 ps	0.04	13 ps	28 ps	37 ps	33 ps
16 ps	-17 ps	0.10	33 ps	20 ps	15 ps	16 ps
16 ps	-3 ps	0.78	19 ps	17 ps	16 ps	16 ps
16 ps	8 ps	0.42	8 ps	14 ps	16 ps	16 ps
16 ps	16 ps	0.12	0 ps	11 ps	17 ps	16 ps
8 ps	-12 ps	0.22	20 ps	11 ps	8 ps	8 ps
8 ps	-5 ps	0.62	13 ps	9 ps	8 ps	8 ps
8 ps	4 ps	0.68	4 ps	6 ps	8 ps	8 ps
8 ps	16 ps	0.12	-8 ps	4 ps	8 ps	8 ps

Table 5.2 – Influence d'une réalisation de *jitter* x d'un étage d'anneau auto-séquencé sur les phases successives autour de l'événement qui la transporte

En observant le tableau, on voit qu'il y a effectivement une dépendance directe entre la réalisation x (qui définit la phase effective $\triangle\varphi_0$) et la prochaine phase théorique $\triangle\varphi_1$. Cependant, elle n'est absolument pas linéaire et semble assez faible. Par exemple (première ligne du tableau), pour une phase moyenne $\triangle\varphi_m = 34$ ps et une phase effective $\triangle\varphi_0 = 51$ ps (correspondant à $x = -17$ ps), la prochaine phase est $\triangle\varphi_1 = 40$ ps puis ensuite $\triangle\varphi_2 = 32$ ps et enfin $\triangle\varphi_3 = \triangle\varphi_m$. On voit que dans la plupart des cas, la phase $\triangle\varphi_2$ vaut presque $\triangle\varphi_m$. De plus les valeurs extrêmes de x sont très improbables, pour de faibles valeurs de $|x|$ (qui sont les plus probables), la phase $\triangle\varphi_2$ vaut exactement $\triangle\varphi_m$ (*c.f.* les deux lignes centrales dans la configuration du milieu avec $\triangle\varphi_m = 16$ ps). Enfin, pour la dernière configuration (avec $\triangle\varphi_m = 8$ ps), la phase $\triangle\varphi_2$ semble complètement indépendante de la réalisation x, ce n'est pas étonnant car dans cette configuration l'influence de l'effet *Charlie* est plus importante (les temps de séparation sont plus petits). Ces mesures semblent cohérentes avec celles présentées dans Tab. 4.1 (section 3.2 du chapitre 4). Finalement, dans ce cas d'étude qui est basé sur des paramètres réels sauf pour l'amplitude du *jitter* (la valeur utilisée est plus grande que dans la réalité), on s'aperçoit qu'il est raisonnable de négliger les dépendances au delà du troisième ordre.

En conclusion, si on néglige les dépendances de $k^{ième}$ ordre dans le STRNG, on peut considérer ses bits de sorties indépendants (et donc $H(B_n|B_{n-1}, ..., B_1, B_0) \simeq H(B_n)$) si au moins l'une des deux conditions suivantes est respectée :

Condition 1 - Le pas d'échantillonnage est différent de D'_m, D_m, $2D'_m$, $D'_m + D_m$, ..., kD'_m, ..., $(k-1)D'_m + D_m$, kD_m, avec $D_m = \frac{N_T T}{2L}$ et $D'_m = \frac{N_T T}{2L} - \triangle\varphi_m$, où $\triangle\varphi_m$ est la résolution de phase moyenne de l'anneau.

Condition 2 - Le pas d'échantillonnage est plus grand que $k \times Dm$ avec $D_m = \frac{N_T T}{2L}$.

En pratique, $k = 3$ voire $k = 4$ semble une valeur raisonnable au vu de nos expériences (permettant de négliger ces dépendances). Cependant nous verrons que dans les implantations réelles, des pas d'échantillonnage aussi rapides ne sont quasiment jamais utilisés (la deuxième

condition est presque toujours respectée avec k grand) : les pas d'échantillonnage considérés sont souvent des dizaines de fois voire des centaines de fois supérieurs à D_m. D'autre part, plus le nombre d'événements est grand (indirectement plus L est grand aussi), plus la première condition est facile à respecter. Si on respecte la condition 2, on diminue aussi le risque de dépendances à long terme entre bits non successifs (qui seraient possibles si on s'assure uniquement de la condition 1 sans que la condition 2 soit garantie). Enfin précisons que si l'échantillonnage est interne, la première condition est systématiquement respectée à partir du moment que $k < N_T$ puisque le pas d'échantillonnage est par défaut égal à T (le même événement est donc ré-échantillonné au bout de N_T cycles).

2.6. Entropie en sortie du filtre de parité

Une modélisation mathématique d'un filtre de parité est présentée dans [Dav02]. Nous rappelons que le principe d'un filtre de parité d'ordre n consiste à sommer n bits en entrée B_{in_0}, B_{in_1}, ..., B_{in_n} (ceux-ci défilent en série à l'entrée du filtre) pour produire un bit de sortie noté B_{out} :

$$B_{out} = B_{in_1} \oplus B_{in_2} \oplus ... \oplus B_{in_n} \tag{5.13}$$

Selon les auteurs de [Dav02], si les bits d'entrée sont indépendants, alors on peut exprimer la probabilité que le bit de sortie B_{out} vaille 'u' ('u' étant égal à '1' ou à '0'), notée $P_{out}(u)$ en fonction de la probabilité $P_{in}(u)$ des bits d'entrée :

$$P_{out}(u) = 0.5 - 2^{n-1}(P_{in}(u) - 0.5)^n \tag{5.14}$$

On remarque que plus n est grand, plus P_{out} s'approche de 0.5, et donc plus l'entropie du bit en sortie du filtre s'approche de 1. L'auteur traite également le cas de bits d'entrée corrélés mais ce calcul n'est pas exploitable avec notre modèle car on ne dispose pas des probabilités conditionnelles $P(B_n|B_{n-1})$. Cette relation n'est donc juste dans notre cas que si les conditions présentées dans la section précédente sur la fréquence d'échantillonnage sont respectées, de manière à garantir l'indépendance des échantillons successifs.

Finalement, le filtre de parité permet de réduire le biais statistique et de relever l'entropie minimale par bit de sortie du STRNG si ceux-ci sont indépendants (en respectant la condition sur le pas d'échantillonnage), en compressant n bits successifs pour produire un bit de sortie (ce qui réduit le débit par n). Le seuil bas d'entropie par bit en sortie du filtre de parité, noté Hf_m, est donc calculé avec le système d'équations suivant :

$$Hf_m = -P_f log_2(P_f) - (1 - P_f) log_2(1 - P_f) \tag{5.15}$$

avec :

$$P_f = 0.5 - 2^{n-1}(P_{in} - 0.5)^n \quad et \quad P_{in} = 1 - 2\Phi(\frac{T}{4L\sigma}) + 2(\Phi(\frac{T}{4L\sigma}))^2 \tag{5.16}$$

2.7. Lien entre le modèle et le matériel

Le seuil bas d'entropie proposé dans ce chapitre est fonction de l'amplitude du *jitter* de l'anneau auto-séquencé σ, de sa période d'oscillation T et de son nombre d'étages L. σ et T peuvent être mesurés pendant la phase de développement du circuit (celle-ci n'est pas spécifique à chaque implantation, mais plutôt à une technologie donnée), alors que L est directement contrôlé par le concepteur. Nous rappelons que la période d'oscillation ne dépend absolument pas du nombre d'étages de l'anneau, mais plutôt de son taux d'occupation (qui peut être maintenu constant en augmentant le nombre d'étages). D'autre part, l'amplitude du *jitter* ne varie pas du tout avec le nombre d'étages. Ces mesures peuvent donc être effectuées sur un anneau quelconque (peut importe sa taille). Cependant, la mesure de *jitter* doit être effectuée avec précaution.

La mesure de la période d'oscillation T correspond à une mesure de moyenne, elle ne présente pas de difficulté particulière. Par contre la mesure de *jitter* est beaucoup plus délicate. D'abord, il est préférable de l'effectuer en implantant uniquement un anneau auto-séquencé dans la cible visée. En effet, l'objectif est de mesurer le *jitter* minimal dû aux étages de l'anneau auto-séquencé. Il faut donc minimiser toutes les influences externes qui peuvent rajouter au bruit mesuré (y compris l'extracteur d'entropie). D'autre part, les mesures effectuées sur le signal de sortie permettent de quantifier le *period jitter* (σ_{period}), qui est certes du même ordre de grandeur, mais légèrement supérieur au *jitter* généré localement dans un étage de l'anneau (σ). Néanmoins, nous avons quantifié ce lien dans le chapitre 4 pour une configuration d'anneau proche de sa fréquence maximale, *i.e* avec un point de fonctionnement proche de la vallée de la courbe *Charlie*, ce qui est typiquement le cas de configurations multiphasées avec un nombre d'étages important (*c.f.* Tab. 4.2 dans la section 3.3 du chapitre 4). L'amplitude du *jitter* généré localement dans un étage de l'anneau peut être estimée à partir du *period jitter* en utilisant la relation suivante (résultat de la section 3 du chapitre 4) :

$$\sigma \simeq \frac{\sigma_{period}}{1.7} \tag{5.17}$$

Cette mesure est d'autant plus délicate et imprécise que l'amplitude du *jitter* est faible. Or, c'est malheureusement le cas généralement pour les anneaux auto-séquencés.

Le seuil d'entropie minimal que nous avons estimé dans ce chapitre est indépendant des instants d'échantillonnage. Le pas d'échantillonnage minimal n'est en pratique limité que par la résolution de phase de l'anneau, celui-ci doit en effet être supérieur à $2\triangle\varphi$ afin de ne pas échantillonner deux fois la même réalisation de *jitter* d'un événement. Cependant, comme nous avons remarqué dans la section 2.5, il faut respecter certaines conditions sur ce pas d'échantillonnage afin d'obtenir des bits successifs indépendants. En pratique, il n'est pas nécessaire de connaître précisément le rapport sauf si on souhaite des débits proches des limites permises par la technologie (*i.e.* les délais des portes logiques de base). Pour avoir une idée de ces grandeurs, prenons l'exemple d'un anneau auto-séquencé de 127 étages avec 64 jetons dans une cible FPGA. Cet anneau a une période autour de 2,5 ns (400 MHz). Si on souhaite éviter les dépendances jusqu'au $k^{ième}$ ordre (condition 2 de la section 2.5), alors il faut que le pas d'échantillonnage soit supérieur à $T_{min} = k \times \frac{64 \times 2.5ns}{2 \times 127}$. Pour $k = 1$, on a $F_{max} = \frac{1}{T_{min}} = 1.59$ GHz. Pour $k = 3$, on a $F_{max} = 530$ MHz. Et pour $k = 10$, on a $F_{max} = 159$ MHz. Nous rappelons que des débits de l'ordre de 1 Mbit/s (qui donnent $k > 100$)

sont largement suffisants pour la plupart des applications [Fis12].

2.8. Stratégies de dimensionnement du générateur

Dans la suite nous supposons que les critères sur la fréquence d'échantillonnage sont respectés : en pratique il n'est pas nécessaire de vérifier cela sauf si les débits envisagés sont de l'ordre de celui de l'anneau auto-séquencé et supérieurs. Le principe de dimensionnement du STRNG consiste à mesurer σ (l'amplitude du *jitter* de l'anneau auto-séquencé) et T (la période de l'anneau) pour un anneau auto-séquencé quelconque (ils ne varient pas significativement avec le nombre d'étages), puis de tracer la courbe représentant le seuil bas d'entropie en fonction du nombre d'étages (comme dans Fig. 5.4). En fonction du niveau de sécurité exigé et des contraintes de taille du *design*, on peut imaginer plusieurs stratégies pour le réglage et dimensionnement du STRNG. Dans cette section, nous proposons d'en analyser quelques unes.

Stratégie du moindre risque Une première stratégie consiste simplement à choisir le nombre d'étages L de manière à ce que $Hm > 0.99$. En appliquant cette stratégie, la sortie du générateur devrait produire en théorie une suite de nombres véritablement aléatoires, indépendants et non biaisés. Les suites générés devraient théoriquement passer la plupart des tests statistiques sans avoir recours à un post-traitement arithmétique. Un post-traitement de type cryptographique peut toutefois être utilisé comme garantie de sécurité supplémentaire en cas de défaillance temporaire de la partie physique. Cependant, d'une part, il n'est pas toujours possible de faire des anneaux auto-séquencés avec un nombre d'étages suffisamment grand pour que $Hm > 0.99$ (par exemple dans des cibles contraintes comme les FPGA). D'autre part, comme nous l'avons remarqué précédemment (dans la section 2.4), pour $\sigma/T = 0.001$, il faut doubler le nombre d'étages pour passer de $Hm = 0.90$ à $Hm > 0.99$. Cette stratégie n'est donc pas rentable du point de vue de la taille et de la consommation du *design*. Nous rappelons aussi qu'il n'est pas nécessaire que l'entropie soit proche de 1 au niveau des nombres aléatoires bruts [KS01], du moment qu'on peut la quantifier, et qu'on prend les mesures adéquates sur le post-traitement arithmétique pour réduire le biais résiduel.

Stratégie du compromis Le nombre d'étages L est choisi de manière à obtenir un seuil bas d'entropie relativement important, mais non nécessairement proche de 1 (par exemple, $Hm > 0.70$). En utilisant Eq. (5.15) et Eq. (5.16), on calcule ensuite n_{min} l'ordre minimal du filtre de parité pour que le seuil bas d'entropie en sortie du filtre soit supérieur à 0.99 ($Hfm > 0.99$). Le débit en sortie est alors divisé par n_{min}, mais la taille du *design* reste dans des limites raisonnables. Ceci permet une adaptabilité du *design* en fonction des contraintes de taille, en ralachant la contrainte sur le débit maximal. Cette stratégie est pertinente étant donné les débits très hauts permis par le STRNG. Cependant, il faut être prudent en l'appliquant, car plus le débit initial est important, plus il risque d'y avoir des corrélations entre bits successifs (si l'on suppose que le rapport entre la fréquence d'échantillonnage et la fréquence de l'anneau n'est pas connu précisément), et moins le post-traitement arithmétique est efficace (Eq. (1.2) est d'ailleurs de moins en moins juste).

Stratégie sans implantation des anneaux Notons qu'il est toujours possible de tracer la courbe d'entropie en faisant des estimations sur σ et T. Si l'on dispose des données techniques de la technologie, par exemple le temps de propagation d'une LUT dans un FPGA (noté D_{LUT}),

alors on peut grossièrement estimer la période minimale de l'anneau auto-séquencé par la relation $T = 4 \times D_{LUT}$ (modèle linéaire qui ne prend pas en compte l'effet *Charlie*). En ce qui concerne l'amplitude du *jitter* σ, une estimation de l'ordre de 1 pour 1000 du délai de propagation typique ($\sigma = 0.001 D_{LUT}$) semble raisonnable d'après nos relevés du chapitre 4. On peut alors tracer la courbe d'entropie en reprenant l'une des deux stratégies de réglage précédentes. Cette stratégie est utile si l'étape de développement spécifique à une technologie donnée n'est pas envisageable pour des raisons techniques, de coût ou de délais.

Stratégie pour des applications à faible niveau de sécurité Enfin, si l'application ne nécessite pas l'imprévisibilité des suites, alors le seul critère est que la sortie post-traitée passe les tests statistiques standards. L est donc simplement choisi en fonction des contraintes de taille, et n l'ordre du filtre est ajusté de manière à passer les tests avec un débit optimal.

3. Sécurisation du générateur

La sécurité du STRNG dépend de l'entropie minimale par bit calculée en relation avec une mesure de la source d'aléa (écart-type du *jitter*), mais aussi des moyens mis en oeuvre pour garantir ce seuil minimal d'entropie malgré d'éventuelles attaques sur le générateur. Cette section analyse les éventuelles menaces et point d'attaque du STRNG et propose des méthodes pour améliorer la sécurité du générateur en contrôlant son fonctionnement à différents niveaux (source d'aléa, extraction d'aléa et séquences de sortie).

3.1. Modèle de menace et contre-mesures

Nous proposons un modèle de menace qui résume les éventuels points d'attaque sur le STRNG. Celui-ci est illustré dans Fig. 5.7. En pratique, nous distinguons trois menaces principales susceptibles d'être exploitées par un attaquant pour fauter le fonctionnement du générateur :

- Les attaques visant à injecter un signal déterministe via la tension d'alimentation (par exemple on peut moduler le signal d'alimentation avec une enveloppe périodique).
- Les attaques par injection de faute, soit par laser ou via des émanations électromagnétiques impulsionnelles (ces attaques sont localisées).
- Les attaques électromagnétiques globales (de type harmonique). Celles-ci pourraient agir sur les fronts de l'anneau (à vérifier) mais surtout sur les bascules et l'arbre de XOR.

Ces attaques pourraient (si leur efficacité est avérée pour le STRNG) fauter le générateur de plusieurs manières :

- Perturber la propagation des événements dans l'anneau auto-séquencé et donc modifier la répartition des phases qui ne serait plus alors uniforme.
- Provoquer un déni de service (en bloquant par exemple l'oscillateur) soit en faisant sortir la tension d'alimentation des limites permettant le fonctionnement des cellules de base, ou bien en injectant un signal électromagnétique harmonique ayant une très grande puissance.
- Injecter ou retirer des jetons si l'attaque est suffisamment localisée (par exemple à l'aide d'un laser).

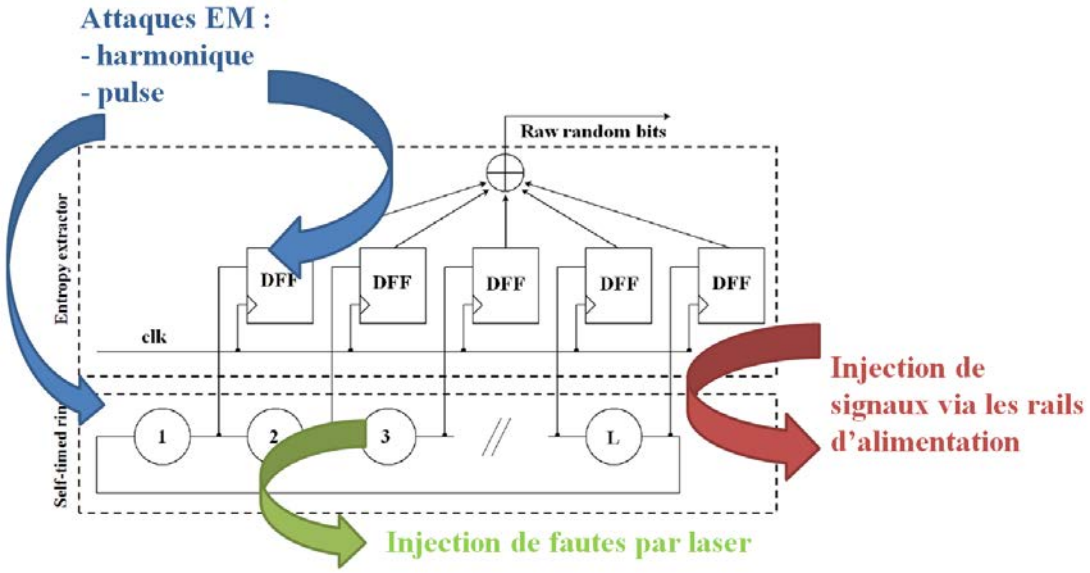


Figure 5.7 – Modèle de menace du STRNG

Les attaques de type déni de service sont *a priori* faciles à détecter via une alarme d'échec total. Il est par exemple assez facile de vérifier si l'oscillateur est bloqué. Les attaques visant à perturber la propagation des événements peuvent également être détectées en contrôlant la période d'oscillation de l'anneau.

Cependant, l'attaque qui nous semble la plus dangereuse est celle visant à injecter ou retirer des jetons. En effet, dans ce cas, l'anneau peut continuer à fonctionner parfaitement normalement ce qui rend difficile de détecter le problème. De plus si l'attaque modifie le nombre de jetons, celui-ci restera le même après que l'attaque soit finie. Ceci est très dangereux, car le réglage des phases de l'anneau dépend directement du rapport N/L (taux d'occupation de l'anneau). En fait il est possible de modifier les phases de l'anneau en modifiant très peu sa période d'oscillation. Prenons l'exemple d'un anneau auto-séquencé à 15 étages. Si l'anneau est initialisé avec 7 événements alors il affichera 15 phases (correspondant au nombre d'étages) car 7 et 15 sont premiers entre eux. Cependant, si on modifie le nombre de jetons à 5, alors certaines phases seront confondues (car 15 est un multiple de 5), la résolution de phase minimale augmente alors de manière drastique (et l'entropie chute).

La principale menace pour le STRNG est donc les attaques visant à modifier sa configuration interne (nombre de jetons). Pour se protéger de ce type d'attaques, il faut contrôler en permanence le nombre de jetons qui circulent dans l'anneau. Fig. 5.8 propose le schéma de principe d'une architecture permettant de réaliser ce contrôle. L'idée consiste à XORer deux-à-deux deux étages successifs (sur tout le long de l'anneau) : si la sortie d'un étage est différente de celle de l'étage suivant (définition d'un jeton), alors le XOR produit un '1', sinon (présence d'une bulle) il produit un '0'. On obtient alors un signal de L bits qui contient autant de '1' que de jetons et de '0' que de bulles. Il suffit alors de mémoriser la sortie des XOR à un instant donné, puis de calculer son poids

de Hamming pour obtenir le nombre de jetons qui circulent dans la structure. Si la valeur détectée est différente de la valeur d'initialisation, alors le générateur a probablement subi une attaque qui a modifié sa configuration interne. Une alarme peut-être alors enclenchée, celle-ci provoque la ré-initialisation du générateur avec sa configuration d'origine.

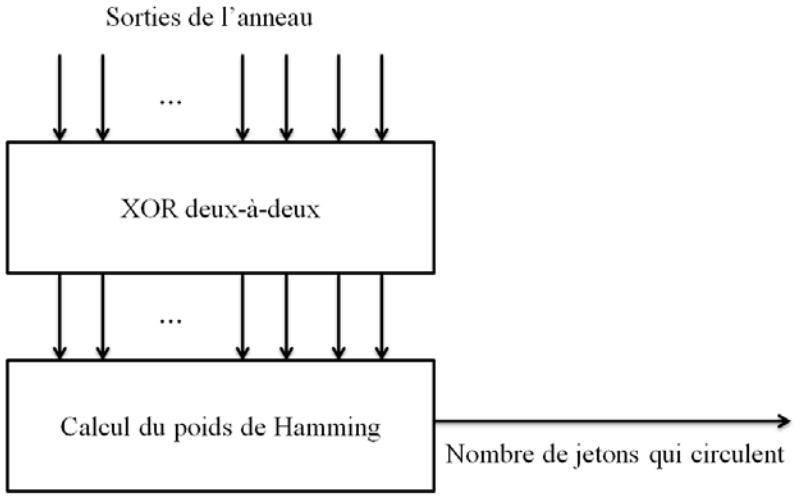


Figure 5.8 – Architecture permettant de contrôler la configuration du STRNG (notamment le taux d'occupation de l'anneau auto-séquencé)

En réalité, il est possible de complètement s'affranchir de ce schéma (relativement lourd) si on respecte une condition très simple au moment du développement du générateur : choisir un nombre d'étages L premier. Ainsi, quelque soit le nombre d'événements N qui circulent, il sera toujours premier avec L, il y aura donc L phases équi-réparties dans l'anneau quelque soit N (du moment qu'on est dans les limites du mode d'oscillation régulier). Dans ce cas, il n'est plus nécessaire de contrôler le nombre de jetons, mais simplement la fréquence d'oscillation de l'anneau. En effet, comme N et L restent premier entre eux, si on est dans le mode régulier alors la relation suivante est valable :

$$\triangle\varphi = \frac{T}{2L}$$

Comme L est figé (nombre d'étages de l'anneau), une modification de N va alors simplement modifier les fréquences (sans compromettre la répartition uniforme des phases), ou au pire des cas provoquer le mode d'oscillation en rafale. Il suffit alors de contrôler la fréquence afin de vérifier que l'anneau n'est pas dans son mode d'oscillation en rafale, et que la résolution de phase est suffisamment fine au vu de l'entropie minimale visée.

3.2. Alarmes et tests spécifiques au principe du générateur

Comme nous l'avons constaté dans la section précédente, le fait de choisir L (le nombre d'étage) premier dès le départ permet une simplification importante de l'architecture permettant le contrôle

de la configuration du TRNG (*i.e* sa répartition de phases et sa résolution temporelle minimale). Cette architecture se base sur un simple fréquencemètre qui doit permettre d'estimer la fréquence moyenne d'oscillation et de détecter si l'oscillation est régulière ou en rafale. Il est possible d'aller plus loin en se servant de cette architecture pour générer diverses alarmes comme cela est montré dans Fig. 5.9 : chaque seuil est associé à une fréquence donnée, elle-même associée à un seuil bas d'entropie donné.

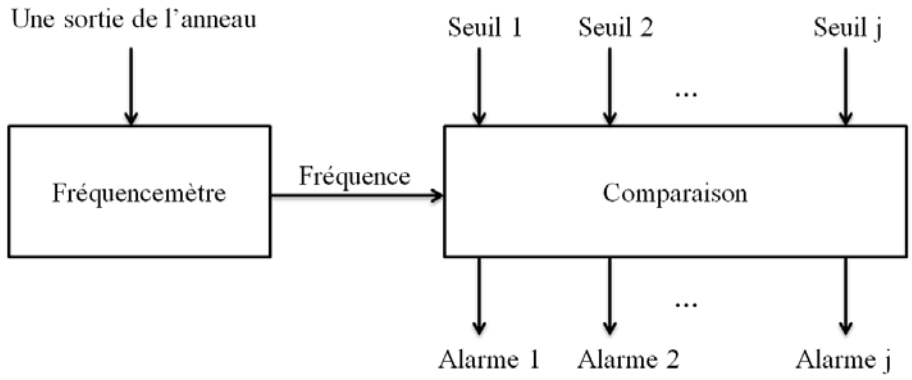


Figure 5.9 – Architecture permettant de contrôler la configuration d'un STRNG utilisant un anneau auto-séquencé avec un nombre d'étages premier

Une alarme d'échec total de la source d'entropie (blocage de l'oscillateur) peut être facilement extraite du module en comparant les valeurs mesurées avec un seuil de fréquence très bas : si la fréquence détectée est trop faible il est possible que le générateur subisse des blocages momentanés (ou il est même complètement bloqué). Ce type d'alarme détecte facilement les attaques par déni de service.

Ensuite, il est possible d'envisager différents seuil d'alarmes qui correspondent à des gammes de résolution de phase (celles-ci dépendent de la fréquence). Comme l'entropie minimale en sortie est directement liée à la résolution de phase, alors ces seuils correspondent littéralement à des "seuils d'entropie minimale". Par exemple, on peut imaginer que pour un *design* donné, le module émette une alarme de basse priorité si l'entropie chute légèrement (*i.e.* quand la fréquence diminue légèrement), une alarme de haute priorité si l'entropie chute de manière importante (*i.e.* quand la fréquence diminue sensiblement), et une alarme critique (alarme d'échec total de la source) si l'oscillateur subit des blocages (même momentanés) ou s'il est dans le mode d'oscillation en rafale. En fonction du niveau de sécurité visé, la détection d'une de ces alarmes devrait enclencher la ré-initialisation du générateur dans sa configuration d'origine. Le protocole de ré-initialisation doit par ailleurs prendre en compte le temps de démarrage de l'anneau auto-séquencé. Si un post-traitement cryptographique est utilisé en conjonction avec le générateur, on peut facilement imaginer un protocole qui puisse stocker des graines aléatoires du STRNG pendant son fonctionnement normal pour que le post-traitement cryptographique assure l'imprévisibilité sur une durée suffisamment longue (le temps de l'attaque, les temps de ré-initialisation ...).

3.3. Mesure embarquée de la source d'aléa

Pour compléter cette revue des méthodes destinées à améliorer la sécurité du générateur, nous proposons des moyens pour monitorer l'entropie en évaluation directement la source la source d'aléa (le *jitter*). Jusqu'à présent, il n'existe pas d'implantation matérielle efficace pour la mesure interne du *jitter*[1]. Dans le cadre de nos travaux, nous avons exploré deux pistes pour la réalisation de cette mesure.

La première technique que nous proposons utilise la résolution de phase d'un deuxième anneau auto-séquencé pour mesurer les périodes de l'anneau utilisé dans le STRNG. Le principe est illustré dans Fig. 5.10. Nous souhaitons mesurer la période du signal r, notée T_O. Pour cela nous exploitons la résolution de phase φ d'un anneau auto-séquencé de L étages initialisé avec un nombre d'événements premier avec L : c'est la résolution temporelle de cette mesure. On sait que :

$$T_{STR} = 2L\varphi \tag{5.18}$$

On compte alors le nombre de fronts de chaque signal C_i (sortie de l'anneau auto-séquencé) pendant la fenêtre T_O. Comme les signaux de sortie de l'anneau (dont on se sert pour effectuer la mesure) sont équi-répartis, alors certains signaux (en fonction de leur phase initiale avec le signal r) vont présenter N fronts montants durant la fenêtre T_O, et d'autres $N+1$ fronts avec :

$$T_O = NT_{STR} + n\varphi \pm \delta \tag{5.19}$$

où n est le nombre de signaux qui affichent N front montant et $\delta < \varphi$. Il suffit alors de déterminer N et n en évaluant les sorties des compteurs pour obtenir une valeur de T_O avec une résolution φ. En répétant l'opération plusieurs fois, on obtient une population de périodes dont on peut évaluer les distributions et les écarts-types.

Nous avons réalisé et testé des implantations de cette technique sur des cibles FPGA. L'acquisition des valeurs de N et n se fait à l'intérieur du circuit, mais le traitement des données est effectué en logiciel. Nous avons constaté que les valeurs mesurées sont beaucoup plus importantes que celles prévues, ce qui était prévisible étant donné que les signaux de l'anneau auto-séquencé (qui permettent la mesure) sont aussi sujet au *jitter* : il faut développer un modèle de calcul qui prenne se facteur en compte, ou à défaut, imaginer une méthode de calibrage. Par exemple nous avons testé cette méthode pour mesurer le *jitter* d'un horloge générée à partir d'un quartz. Nous avons trouvé une distribution gaussienne mais des valeurs d'écart-type deux fois supérieures à celles mesurées à l'oscilloscope. Au final nous n'arrivons donc pas à estimer précisément la valeur de *jitter*, nous pouvons juste déterminer si elle augmente ou elle diminue, ou bien la comparer pour deux oscillateurs. D'autre part, l'implantation est très couteuse et présente des défauts importants qui semblent fausser les mesures. En particulier la gestion du *skew* des signaux qui alimentent les bascules n'est pas du temps prise en compte dans l'implantation car il y a beaucoup plus d'horloges que ne le permet la circuiterie d'arbre d'horloge du FPGA, ce qui semble dégrader la répartition

1. Il existe des travaux qui effectuent des mesures à l'intérieur d'un circuit avec des techniques d'échantillonnage cohérent utilisant deux oscillateurs en anneaux, cependant le traitement des données est effectué à l'extérieur du circuit

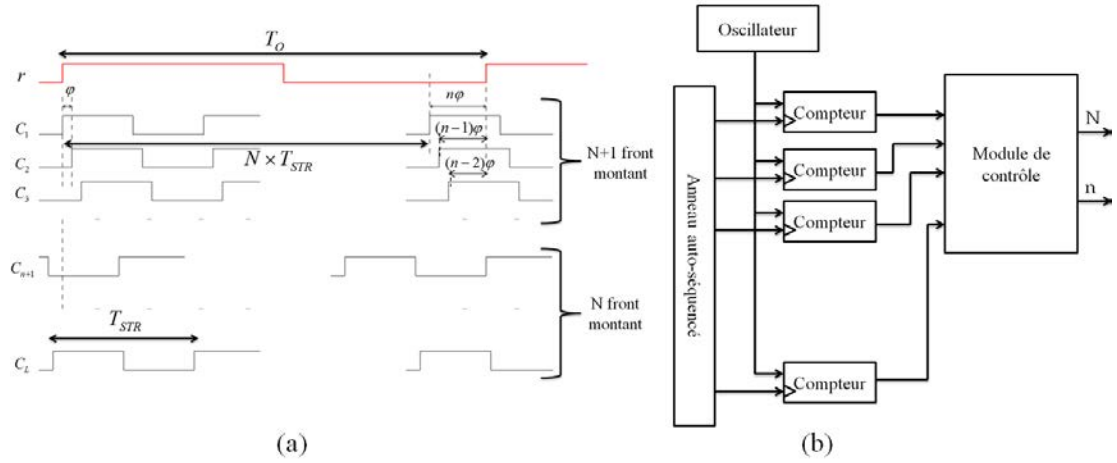


Figure 5.10 – Mesure de la période d'un oscillateur en utilisant la résolution de phase d'un anneau auto-séquencé : (a) principe (b) architecture

des signaux aux entrées des compteurs. Pour conclure sur cette technique, le principe semble très prometteur car il permet une résolution de mesure très fine, mais nous n'avons pas encore développé une architecture efficace ainsi que les modèles pour l'exploiter judicieusement dans le cadre de la mesure embarquée du *jitter*.

La deuxième technique que nous proposons pour monitorer l'entropie est beaucoup moins précise en théorie (de loin) mais elle est plus facilement réalisable dans le matériel, et elle ne nécessite pas de traiter les données en logiciel. Ce n'est pas une mesure de *jitter* à proprement dit, mais plutôt une évaluation simple de sa valeur relative à la résolution de phase du STRNG. Elle nécessite cependant de basculer dans le mode interne du générateur (sans forcément arrêter le *bit-stream*). Pour rappel, dans le mode interne, le signal d'échantillonnage est issu de l'anneau auto-séquencé : il est synchronisé avec les signaux qu'il échantillonne (sorties des étages de l'anneau). Dans ce cas, si on observe les sorties des bascules, celles-ci vont varier plus ou moins fréquemment en fonction de la phase entre le signal échantillonnant et le signal échantillonné, et certaines bascules vont toujours afficher la même valeur. On évaluant le nombre de bascules dont les valeurs changent (avec une fréquence minimale donnée), on obtient directement une indication sur la taille relative du *jitter* par rapport au réglage des phases : plus les bascules qui varient sont nombreuses, plus ce réglage est fin. Les alarmes peuvent être calibrés par rapport à des configurations connues de $\sigma/\triangle\varphi$.

Le principe est illustré dans Fig. 5.11. Les valeurs successives des sorties des bascules sont mémorisées dans des registres (ces deux mémorisations sont séparées d'un temps T_{eval}). On calcule les distances de Hamming entre les vecteurs successifs (constitués des bits de sorties des bascules) pour savoir le nombre de bascules dont la valeur a changé. Ces valeurs sont comparées à des seuils pré-établis pour évaluer la taille du *jitter* par rapport à la résolution de phase de l'anneau. Les seuils d'alarmes et valeurs de monitoring peuvent être calibrés pendant la phase de développement du générateur (pour un temps T_{eval} donné) et correspondent à différentes valeurs d'entropie liées

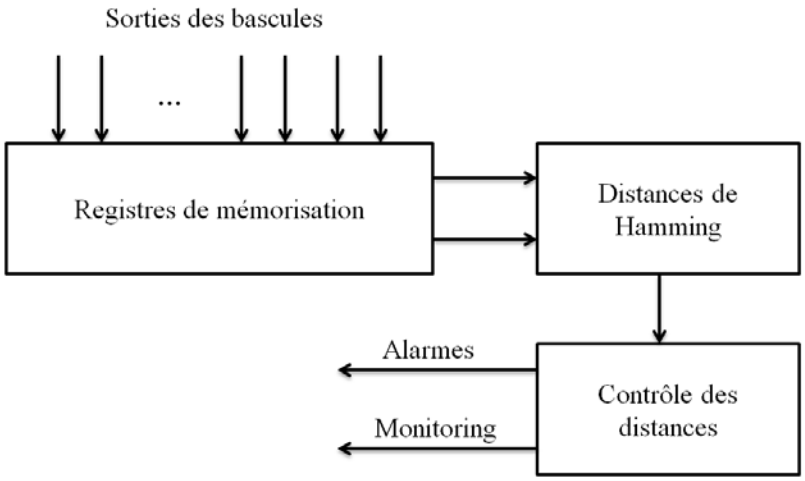


Figure 5.11 – Architecture permettant de contrôler l'entropie en sortie d'un STRNG

au rapport $\sigma/\triangle\varphi$. Cette étape de calibrage est spécifique à une technologie donnée (et non à chaque implantation) tout comme l'étape de dimensionnement du générateur.

4. Conclusion

Dans ce chapitre, nous avons estimé un seuil bas d'entropie par bit de sortie du générateur en fonction de l'amplitude du *jitter* σ, de la période d'oscillation de l'anneau T et de son nombre d'étages L. Cette relation est donnée par Eq. (5.12) et Eq. (5.11) :

$$H_m = -P(u)_{t=0} log_2(P(u)_{t=0}) - (1 - P(u)_{t=0}) log_2(1 - P(u)_{t=0})$$

avec :

$$P(u)_{t=0} = 1 - 2\Phi(\tfrac{T}{4L\sigma}) + 2(\Phi(\tfrac{T}{4L\sigma}))^2$$

Le réglage de ce seuil bas d'entropie s'effectue en mesurant σ et T, puis en choisissant L en fonction du niveau de sécurité désiré. Nous avons donc proposé des stratégies de dimensionnement du STRNG (exploitant ce modèle) en fonction du niveau de sécurité exigé par l'application et des contraintes de taille imposées par la cible d'implantation.

Dans la section 2.5, nous avons dérivé les conditions sur la fréquence d'échantillonnage pour obtenir des bits indépendants entre eux en sortie du générateur. Afin d'éviter d'obtenir deux bits successifs avec une dépendance au dessous du $k^{ième}$ ordre (comme cela est défini dans la section 2.5), il suffit que le pas d'échantillonnage soit supérieur à $k \times \frac{N_T T}{2L}$, où N_T est le nombre de jetons et $\triangle\varphi_m = T/2L$ la résolution de phase moyenne de l'anneau. Des résultats de simulation numérique montrent qu'une dépendance au delà du $3^{ième}$ ordre peut raisonnablement être négligée. D'autre part, nous constatons qu'au vu des débits envisagés cette condition n'est pas contraignante en pratique.

Enfin, nous nous sommes intéressés au facteurs qui pourraient dégrader le fonctionnement du générateur et nous avons proposé une série de mesures pour détecter d'éventuels dysfonctionnements. Le principal point d'attaque du STRNG est, de notre point de vue, sa configuration en nombre de jetons : en injectant ou en retirant des jetons il est possible de dégrader sa résolution de phase (et donc de faire chuter l'entropie), il faut alors contrôler en permanence le nombre de jetons qui circulent. Cependant, nous montrons qu'en choisissant un nombre d'étages L premier dès le départ, alors ce contrôle se résume à la fréquence d'oscillation uniquement (il faut juste contrôler qu'elle ne descend pas au dessous d'un certain seuil). Enfin, pour un STRNG dont le nombre d'étages est fixé, l'entropie dépend directement du rapport $\sigma/\triangle\varphi$ (amplitude du *jitter* sur la période d'oscillation de l'anneau). Nous proposons donc un moyen simple d'estimer ce rapport en évaluant le nombre et la fréquence à laquelle varient les bascules de sortie dans le mode interne.

Chapitre 6.

Evaluation du générateur dans des cibles ASIC et FPGA

Sommaire

1. Introduction

Pour compléter l'étude sur le TRNG à base d'anneau auto-séquencé, nous proposons dans ce chapitre d'évaluer le coeur du générateur dans des circuits reprogrammables (FPGA) et des circuits intégrés (ASIC).

La première partie de ce chapitre s'intéresse à l'implantation du STRNG. Elle présente les conditions à respecter pour une implantation conforme du STRNG et propose une méthode d'implantation dans des cibles FPGA. Par ailleurs, nous avons réalisé un circuit prototype dans une technologie CMOS 350 nm. A travers cette réalisation, nous illustrons la méthode de dimensionnement utilisant le modèle et les mesures présentée dans le chapitre précédent.

La deuxième partie du chapitre évalue le coeur du générateur dans ces cibles d'implantation selon deux critères : l'imprévisibilité de sa sortie (le seuil minimal d'entropie par bit calculé à l'aide des mesures du chapitre 4 et du modèle du chapitre 5), et aussi ses propriétés statistiques (à l'aide des batteries de tests statistiques FIPS 140-1, AIS31 et NIST SP 800-22).

2. Implantation du générateur

Cette section décrit l'implantation du STRNG dans des cibles ASIC et FPGA.

2.1. Contraintes d'implantation

Implantation des anneaux Les contraintes d'implantation des anneaux auto-séquencés sont présentées dans la section 4.1 du chapitre 4. La contrainte principale est que les étages de l'anneau doivent avoir des caractéristiques temporelles proches. En effet, la présence d'un délai très long peut causer un phénomène de *bottleneck* comme le montre la section 4.4 du chapitre 4, celui-ci dégrade fortement la répartition des phases autour de l'étage concerné. Dans ce cas, bien qu'on conserve autant de phases différentes que dans une configuration sans *bottleneck*, celles-ci ne sont plus uniformément réparties. Cette contrainte ne pose généralement pas de problème pour les cibles ASIC. Par contre, pour les implantations FPGA, il est parfois difficile de précisément maîtriser le placement et le routage, surtout pour les anneaux longs (à partir d'une centaine d'éléments).

Distribution des phases à l'entrée des bascules Les signaux en sortie de l'anneau auto-séquencé présentent une distribution de phases uniformes. Cependant, si les délais entre les étages de l'anneau et les entrées de bascules sont différents, alors la répartition des phases aux entrées des bascules n'est plus nécessairement uniforme. Il faut donc que ces délais soient similaires. Ceci peut être réalisé en se servant d'un étage de l'anneau et de la bascule correspondante comme une cellule de base (par exemple en répétant le même motif au niveau du *layout*).

Clock skew Les système synchrones utilisent généralement un arbre de distribution d'horloge qui permet de limiter le retard du signal d'horloge qui arrive en entrée des différentes bascules (*clock skew*). Ce facteur doit aussi être pris en compte dans la conception du STRNG (il faut prévoir un arbre de distribution d'horloge). En effet, le principe du STRNG stipule que quelque soit l'instant d'échantillonnage t, il existe un étage j qui bascule à un moment t_j tel que $|t - t_j| \leq \frac{\triangle\varphi}{2}$. En

supposant que les temps effectifs d'arrivée des signaux d'horloges en entrée des L bascules soient notés t_1, t_2, ..., t_L, alors il faut que tous ces temps soient compris dans un intervalle de longueur $\triangle\varphi$. Si on note D_{skew_i} le *skew* associé au signal d'horloge de la bascule i, alors la contrainte est la suivante :

$$Max(|D_{skew_i} - D_{skew_j}|)_{1 \leq i,j \leq L} \leq \triangle\varphi \tag{6.1}$$

En pratique, cette condition est de plus en plus difficile à respecter quand le nombre d'étages augmente (et donc la résolution de phase diminue).

2.2. Implantation dans des cibles FPGA

Les circuits reprogrammables ont des fonctionnalités plus limitées que les circuits ASIC (en terme de maîtrise du *design*), mais ils sont très utiles à la fois pour le prototypage, et aussi pour les applications qui nécessitent une mise à jour régulière des fonctions implantées. Dans cette section, nous présentons une implantation du STRNG dans deux cibles FPGA : Altera Cyclone III et Xilinx Virtex 5.

L'architecture du générateur est donnée dans Fig. 3.2. Les anneaux sont implantés selon la méthode décrite dans la section 4.1 du chapitre 4. Les anneaux implantés dans Altera ne sont pas reconfigurables de manière dynamique (il faut re-progammer la carte pour changer la configuration) car elles utilisent des LUT à 4 entrées (on peut uniquement avoir un signal de *set* ou un signal de *reset*). Par contre, les anneaux implantés dans Virtex 5 utilisent des LUT à 5 entrées, ils sont donc reconfigurables de manière dynamique (chaque étage possède un signal de *set* et de *reset*).

Chaque étage de l'anneau est implanté dans une LUT. Afin de respecter la contrainte sur la distribution des phases à l'entrée des bascules, nous utilisons la connexion directe (qui ne traverse pas le réseau de routage du FPGA) entre chaque LUT et la bascule correspondante comme cela est montré dans Fig. 6.1 (des connexions directes entre LUT et bascules sont disponibles dans la plupart des FPGA récents). Le signal d'horloge global est défini à la compilation ce qui permet de distribuer ce signal via l'arbre d'horloge dédié du FPGA afin de limiter le *skew*. Dans Altera (respectivement Xilinx), l'arbre de XOR consiste en des rangées de LUT à 4 entrées (respectivement 5 entrées) séparées par des rangées de bascules (pour minimiser le temps critique de la structure et permettre des débits de fonctionnement optimaux).

Dans nos premières implantations, nous avons utilisé un filtre de parité matériel tel que décrit dans Fig. 3.3. Par la suite, nous avons réalisé qu'une implantation logicielle est à la fois plus pratique pour nos expériences, mais aussi plus juste pour l'évaluation puisque le post-traitement arithmétique est en théorie une fonction mathématique pure, alors que toute implantation matérielle d'une fonction quelconque contient une part (même extrêmement infime) d'imprévisibilité due aux phénomènes physiques.

Enfin, l'acquisition des bits de sortie se fait grâce au module de transfert USB fourni par la plateforme *Evariste* [FBH13], développée par l'équipe SES au laboratoire Hubert Curien.

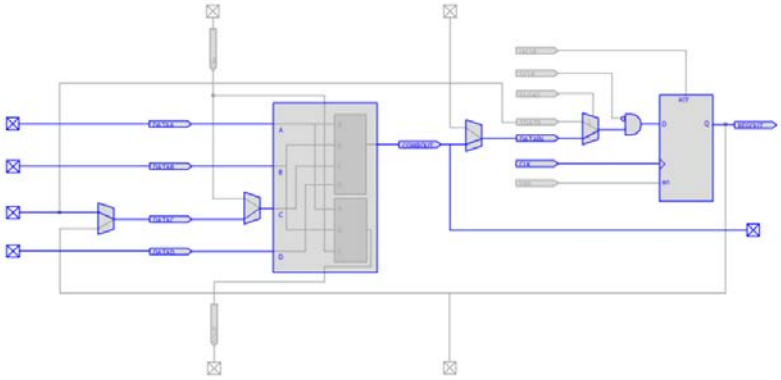


Figure 6.1 – Implantation d'un étage de l'anneau auto-séquencé et de la bascule correspondante dans un bloc logique dans Altera Cyclone III

2.3. Dimensionnement et implantation d'un circuit prototype en technologie CMOS 350 nm

De manière générale, il est plus facile de maîtriser les paramètres du *design* dans les ASIC (et donc de respecter les contraintes de la section 2.1). Cependant, dans le cas des TRNG, le concepteur n'a pas droit à l'erreur car le fonctionnement d'un TRNG ne peut pas être validé par la simulation électrique comme cela est le cas pour les systèmes classiques. Or, les délais de développement et de production en fonderie sont souvent longs. Ceci justifie grandement la démarche que nous utilisons pour le STRNG, basée sur la mesure des paramètres de la source pour dimensionner le générateur.

Pour dimensionner ce générateur, nous nous sommes basés sur les mesures du chapitre 4, qui ont été réalisées dans un premier circuit contenant des anneaux auto-séquencés dans la même technologie ciblée (CMOS 350 nm). Ce principe de dimensionnement est développé dans les sections 2.7 et 2.8 du chapitre 5, il utilise Eq. 5.11 pour estimer un seuil bas d'entropie en fonction de L (nombre d'étages de l'anneau) pour un rapport σ/T donné (amplitude du *jitter* sur la période d'oscillation de l'anneau).

Les mesures de *jitter* pour cette technologie sont donnés dans Tab. 4.5. Nous utilisons la colonne des valeurs pire cas du *period jitter* (σ_p) correspondant à l'approximation en loi linéaire. Cette valeur ne varie pas de manière significative avec le nombre d'étages, elle vaut environ $\sigma_p = 4$ ps. La section 2.7 du chapitre 5 fait le lien entre les valeurs mesurées sur la période d'oscillation σ_p et la valeur à appliquer pour le modèle σ (qui correspond au *jitter* généré localement dans un étage de l'anneau). Ce lien est donné par Eq. 5.17. En appliquant cette formule, on trouve $\sigma = 2.35$ ps. Nous estimons grossièrement la période d'oscillation à $T = 2$ ns (le tableau montre que la fréquence varie entre 469 Mhz et 592 Mhz lorsque le taux d'occupation est optimal). On peut donc tracer la courbe qui donne une estimation de l'entropie minimale par bit de sortie en fonction du nombre d'étages L, celle-ci est donnée dans Fig. 6.2.

Pour le choix de L, nous souhaitons avoir un taux d'entropie raisonnable tout en respectant une contrainte de taille d'environ 200 étages (nous disposons d'un espace limité dans le circuit qui

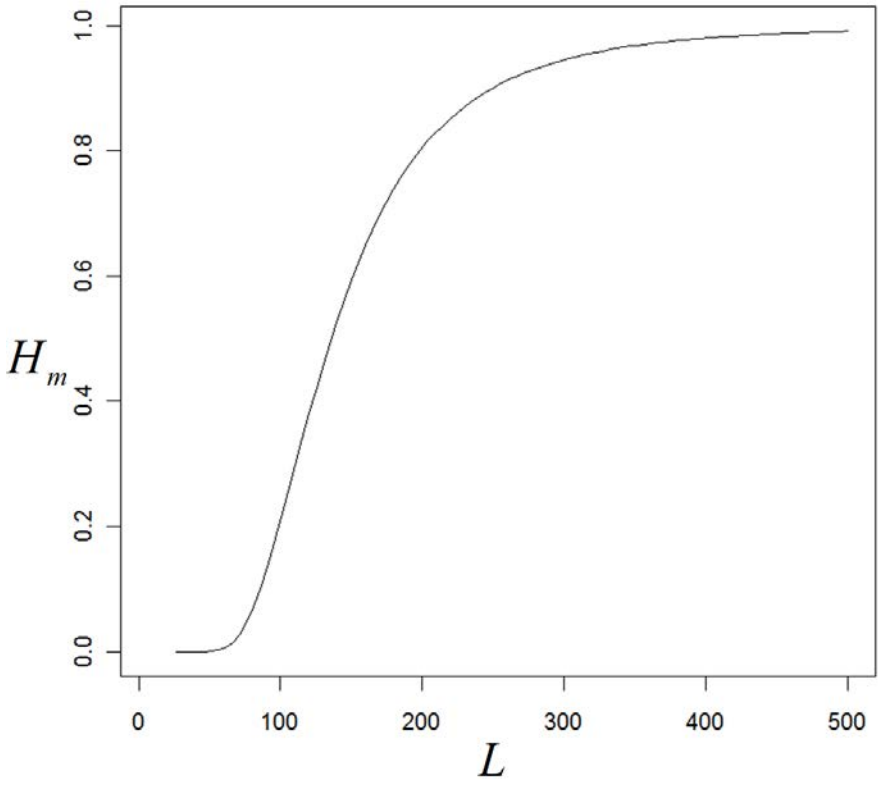


Figure 6.2 – Estimation du seuil bas d'entropie par bit du sortie du STRNG, en fonction de son nombre d'étages L, pour la technologie CMOS 350 nm ($\sigma/T \simeq 0.0012$)

contient d'autres modules associés à d'autres projets réalisés dans l'équipe et qui ne font pas partie de ce travail de thèse). Nous appliquons la stratégie basée sur le compromis, présentée dans la section 2.7 du chapitre 5. Selon la courbe de Fig. 6.2, il faudrait environ 600 étages pour un taux d'entropie supérieur à 0.99. Notre choix s'est arrêté à $L = 163$, ce qui donne un taux d'entropie minimal par bit $Hm = 0.67$. Par ailleurs, 163 est un nombre premier, cela veut dire que toutes les configurations dans le mode régulier sont des configurations multiphasées (avec autant de phases différentes que le nombre d'étages). Etant donné ce taux d'entropie, nous calculons n_{min} l'ordre minimal du filtre pour éliminer le biais résiduel et afin d'obtenir $H_m = 0.99$ en sortie du post-traitement arithmétique (*c.f.* la section 2.6 du chapitre 5). Nous trouvons $n_{min} = 6$, il s'agit de l'ordre du filtre qui devrait théoriquement être implanté. Dans le cadre de ce prototype, nous utilisons un filtre de parité logiciel pour nos besoins d'évaluation (cependant nous verrons dans les résultats d'évaluation que nous n'aurons pas du tout besoin de l'utiliser).

L'implantation des anneaux auto-séquencés est décrite dans la section 5.1 du chapitre 4. Afin de respecter la condition sur la distribution des phases à l'entrée des bascules, nous avons réalisé le *layout* d'une cellule de base composée d'un étage de l'anneau auto-séquencé et de la bascule correspondante. Puis nous avons reproduit le même motif pour tous les autres étages. Ceux-ci sont placés selon une topologie rectangulaire. Un arbre d'horloge a été réalisé (par l'outil automatique)

afin de distribuer le signal d'horloge sur les entrées des différentes bascules en limitant le *skew*. L'arbre de XOR a été réalisé de la même manière que dans Altera Cyclone III, en utilisant des portes XOR à 4 entrées, séparées de rangée de bascules (le *layout* a été fait par l'outil automatique).

L'architecture implantée est donnée dans Fig. 6.3. Le signal *init* initialise le générateur. Le signal $config$ permet de modifier la configuration en jetons de l'anneau à 163 étages. Un multiplexeur est utilisé pour basculer du mode de fonctionnement externe (utilisant une horloge externe réglable) et le mode de fonctionnement interne (utilisant un signal issu de l'anneau). Au moment de la conception de ce circuit, le module de transfert n'avait pas été validé pour des débits très hauts. Nous avions donc divisé le signal de l'anneau pour le mode interne de manière à limiter le débit à 2 Mbit/s (le mode externe ne pose pas de problème car l'horloge externe est réglable).

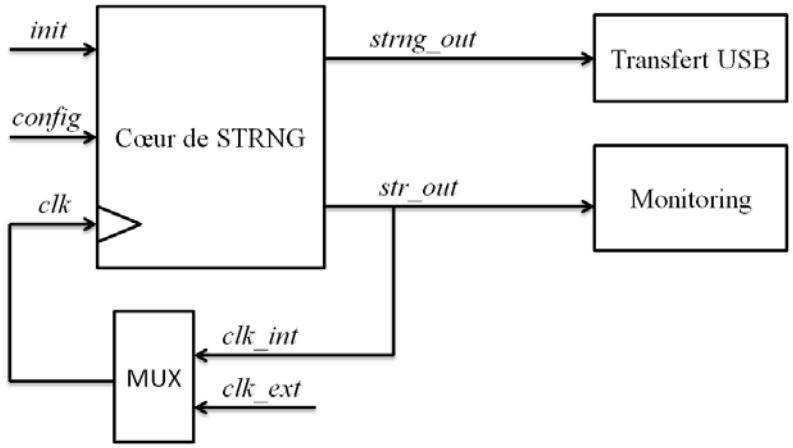


Figure 6.3 – Architecture du circuit réalisé en technologie CMOS 350 nm

3. Evaluation du coeur du générateur

Dans cette section, nous évaluons le coeur du générateur selon deux critères : l'imprévisibilité de sa sortie (l'entropie par bit) et ses propriétés statistiques (à l'aide de batteries de tests standard).

3.1. Estimation d'entropie

Les estimations d'entropie données dans cette section se basent sur le modèle présenté dans le chapitre 5 en utilisant les mesures du chapitre 4. Comme nous l'avons précisé dans la section 2.7 du chapitre 5, il faut réaliser la mesure du *jitter* en implantant uniquement les anneaux dans un premier temps (pour mesurer le *jitter* minimal dans la cible). Les valeurs de σ_p (*period jitter*) sont mesurées dans le chapitre 4 pour les cibles FPGA, elles valent environ 3 ps pour Altera et 4 ps pour Xilinx. Ce qui nous donne $\sigma = 1.77$ ps sur Altera et $\sigma = 2.35$ ps pour Xilinx. En ce qui concerne les fréquences, elles sont au contraire mesurées pendant le fonctionnement du générateur

(ce qui correspond à la situation réelle de charge des cellules). Ces mesures sont données dans Tab. 6.1.

Cible	Configuration		Mesures		Modèle	
	L	N	T	$\triangle\varphi$	H_m	n_{min}
Cyclone III	63	32	2.44 ns	19.3 ps	0.00	-
	127	64	3.11 ns	12.2 ps	0.01	> 1000
	255	128	2.93 ns	5.7 ps	0.47	10
	511	256	3.31 ns	3.2 ps	0.88	3
Virtex 5	63	32	2.82 ns	22.4 ps	0.00	-
	127	64	3.23 ns	12.7 ps	0.05	155
	255	128	3.27 ns	6.4 ps	0.60	6
	511	256	3.42 ns	3.3 ps	0.95	2

Table 6.1 – Période d'oscillation (T), résolution de phase ($\triangle\varphi$), seuil bas d'entropie par bit de sortie (H_m), et ordre du filtre minimal pour obtenir $H_m = 0.99$ (n_{min}) pour différentes configurations d'anneaux auto-séquencés dans des cibles Altera Cyclone III and Xilinx Virtex 5

On remarque que pour obtenir des taux d'entropie intéressants, il faut utiliser des anneaux assez longs (au delà de 255 étages). L'utilisation du filtre de parité pour compresser les données et augmenter l'entropie par bit n'est efficace que si les valeurs initiales d'entropie sont suffisamment élevées. Par exemple, pour un anneau à 127 étages, il faudrait utiliser un ordre de filtre supérieur à 1000 dans Altera, alors qu'un ordre égal à 3 est suffisant si le nombre d'étages est 511.

En ce qui concerne l'implantation ASIC, cette estimation d'entropie a été effectuée pendant la phase développement (voir la section 2.3). Pour le STRNG à 163 étages, nous avons calculé les valeurs suivantes :

$$H_m = 0.67 \quad \text{et} \quad n_{min} = 6$$

3.2. Evaluation statistique

Dans cette sous-section, nous analysons les propriétés statistiques de séquences issues d'implantations du STRNG dans des cibles ASIC et FPGA à l'aide de batteries de tests statistiques standard.

3.2.1. Cibles FPGA

Nous utilisons les batteries de tests statistiques FIPS 140-1 et NIST SP 800-22 pour évaluer les suites issues de STRNG implantés dans des FPGA Altera Cyclone III et Xilinx Virtex 5. L'horloge d'échantillonnage est externe (elle provient d'un quartz) et fonctionne à 16 MHz. L'ordre du filtre de parité est fixé à $n = 8$, ce qui donne un débit effectif de 2 Mbit/s pour les bits non post-traités. Notons que ce débit est largement suffisant pour la plupart des applications [Fis12].

Pour chaque configuration, nous évaluons 1000 séquences de 20000 bits à l'aide des tests FIPS 140-1, puis nous relevons le pourcentage de passage de ces tests. Ensuite, nous appliquons les tests NIST SP 800-22 sur 1000 séquences de 10^6 bits avec un seuil de confiance de 0.01. Tab. 6.2 donne les résultats de ces tests pour différentes configurations de STRNG avec ou sans post-traitement (filtre de parité d'ordre 8). Le débit est de 16 Mbit/s pour les configurations sans post-traitement, et de 8 Mbit/s pour les configurations avec post-traitement.

Post-traitement	Configuration		Cyclone III		Virtex 5	
	L	N	FIPS	NIST	FIPS	NIST
sans	63	32	55%	-	14%	-
	127	64	98%	-	1%	-
	128	64	0%	-	0%	-
	255	128	100%	-	99%	-
	511	256	100%	P	100%	-
avec	63	32	100%	-	100%	-
	127	64	100%	P	100%	-
	128	64	0%	-	0%	-
	255	128	100%	P	100%	P
	511	256	100%	P	100%	P

Table 6.2 – Résultats des tests FIPS 140-1 et NIST SP 800-22 pour différentes configurations de STRNG dans le mode externe à 16 Mbit/s (2 Mbit/s pour les nombres post-traités) dans des cibles FPGA (P=le test passe)

Pour les configurations qui n'utilisent pas de post-traitement, les tests FIPS passent systématiquement à partir de $L = 255$ pour les deux cibles utilisées. Cependant, nous observons une meilleure qualité statistique pour les configurations implantées dans Altera Cyclone III. D'ailleurs les tests NIST passent pour $L = 511$ dans Altera (toujours sans post-traitement) mais pas dans Virtex 5. En observant les séquences de près, nous avons remarqué la présence d'un biais important et qui ne va que dans un sens (le générateur produit plus de '1' que de '0' où l'inverse en fonction du vecteur d'initialisation). Nous suspectons que ce biais provient de défauts électriques : nous observons une rapport cyclique qui n'est pas rigoureusement égal à 50% dans ces implantations. Ce phénomène semble plus accentué dans les configurations implantées dans Virtex 5. Cependant, ce biais peut facilement être corrigé à l'aide du post-traitement arithmétique. Afin d'illustrer l'importance du réglage de phase, nous évaluons une configuration avec 128 étages et 64 jetons (N et L ne sont pas premiers entre eux), en ajoutant seulement un étage à la configuration précédente. Cette configuration échoue systématiquement à tous les tests. Ceci montre que l'unique pseudo-aléa dû à la déviation de phase de l'anneau avec l'horloge d'échantillonnage n'est pas suffisant pour produire des séquences qui passent les tests statistiques avec ces débits (ici l'entropie est proche de 0).

Pour les configurations utilisant le post-traitement, les tests FIPS passent systématiquement à partir de $L = 63$ étages pour les deux cibles. Les tests NIST passent à partir de $L = 127$ pour Altera et $L = 255$ pour Virtex. Cependant, nous précisons que l'unique évaluation statistique n'est pas suffisante pour valider une configuration donnée, il faut aussi que le seuil bas d'entropie

soit suffisamment important. Dans la configuration $L = 63$, le taux d'entropie minimal par bit estimé par le modèle est proche de 0, mais cela ne veut pas dire non plus que tous les bits ont une entropie nulle. En effet, comme l'échantillonnage est externe, l'entropie effective de chaque bit dépend du moment d'échantillonnage comme ceci est montré dans Fig. 5.3 du chapitre 5 (bien qu'elle reste toujours supérieure au seuil bas donné par le modèle), certains bits peuvent même avoir une entropie égale à 1. Enfin, la configuration $L = 128$ échoue systématiquement à tous les tests même avec le post-traitement.

Nous avons souhaité évaluer la sortie du générateur à des débits plus importants (400 Mbit/s), même si en pratique ces débits sont rarement nécessaires. Notons qu'à ce débit (qui est plus rapide que la fréquence de l'anneau auto-séquencé), les dépendances entre bits successifs peuvent commencer à apparaître selon l'analyse de la section 2.5 du chapitre 5. En plus des tests NIST SP 800-22 et FIPS 140-1, nous utilisons les tests T5 à T8 d'AIS31 sur un peu plus de 4 Mo de données. Ceux-ci correspondent aux tests de la procédure pour évaluer la sortie brute (T6 à T8) auquel nous avons ajouté le test T5 d'auto-corrélation (issu de la procédure pour tester les nombres post-traités), nous relevons le nombre de tests qui passent parmi ces quatre tests. Les résultats de ces tests sont donnés dans Tab. 6.3. n_p correspond à l'ordre du filtre que nous utilisons en pratique de manière à ce que tous ces tests réussissent. Il est intéressant de le comparer à n_{min} l'ordre du filtre que nous avons calculé en pratique pour que l'entropie minimale en sortie soit supérieure à 0.99.

Cible	Configuration		Entropie		Données brutes		Données post-traitées	
	L	N	H_m	n_{min}	FIPS	T5-T8	n_p	Débit
Cyclone III	63	32	0	-	0%	0/4	7	57 Mbit/s
	127	64	0.01	>1000	0%	0/4	4	100 Mbit/s
	128	64	0	-	0%	0/4	-	-
	255	128	0.47	10	45%	1/4	2	200 Mbit/s
	511	256	0.88	3	99%	3/4	2	200 Mbit/s
Virtex 5	63	32	0	-	0%	0/4	8	50 Mbit/s
	127	64	0.05	155	10%	1/4	3	133 Mbit/s
	255	128	0.60	6	58%	2/4	2	200 Mbit/s
	511	256	0.95	2	61%	3/4	2	200 Mbit/s

Table 6.3 – Seuil bas d'entropie (H_m), ordre minimal du filtre de parité pour obtenir $H_m = 0.99$ en sortie (n_{min}), taux de passage des tests FIPS et des tests T5-T8 pour les données brutes, ordre du filtre utilisé en pratique pour que l'intégralité des tests passent (n_p) et débit effectif après l'application du post-traitement

La qualité statistique est clairement moins bonne que pour un débit de 16 Mbit/s. Aucune configuration ne passe l'intégralité des tests sans post-traitement. Par contre, encore une fois, le filtre de parité corrige facilement ces défauts. Par exemple, les tests FIPS et T5-T8 passent pour $L = 63$ avec un filtre d'ordre 7 dans Altera (ce qui donne un débit de 57 Mbit/s en sortie du post-traitement). Pour $L = 511$, un filtre d'ordre 2 est suffisant pour passer ces tests (avec un débit de

200 Mbit/s). En ce qui concerne la configuration avec 128 étages et 64 jetons, même avec un ordre supérieur à 20 aucun test ne passe. Enfin, les test NIST SP 800-22 (qui ne sont pas représentés dans le tableau) passent pour Altera pour une configuration $L = 511$ avec un filtre d'ordre 3 (133 Mbit/s). Ils passent pour la même configuration dans Xilinx avec un filtre d'ordre 4 (100 Mbit/s). Notons que même si les taux d'entropie sont plus importants pour Xilinx, les défauts électriques dans le rapport cyclique des signaux semblent plus accentués. La correction doit donc être plus importante.

Enfin, en fonction du niveau de sécurité visé, il faut à la fois prendre en compte le taux d'entropie estimé et les résultats statistiques pour construire un STRNG. Si on reprend les critères de la section 4.6 du chapitre 1 (classes de TRNG selon AIS31), pour construire un TRNG visant la classe RNG. 1, nous recommandons un configuration à 127 étages, avec un filtre de parité d'ordre 4, et un débit fonctionnel de 100 Mbit/s. Pour construire un TRNG visant la classe RNG. 2 ou RNG. 3, il faut que l'entropie minimale par bit soit suffisamment importante au vu du modèle et de la mesure, les configurations qui passent tous les tests mais qui ont un faible taux d'entropie sont à exclure. Le nombre d'étages à utiliser pour viser cette classe de générateur serait typiquement autour de 511 étages, avec l'utilisation d'un filtre de parité d'ordre 2 au minimum.

3.2.2. Circuit prototype en technologie CMOS 350 nm

Pour le circuit CMOS 350 nm, nous avons appliqué les deux procédures d'AIS31 (pour les nombres bruts et post-traités) directement sur la sortie brute du générateur (environ 4.5 Mo de données). Le résultat de chaque test est donné séparément dans Tab. 6.4 pour le mode interne avec un débit de 2 Mbit/s, et dans Tab. 6.5 pour le mode externe avec un débit de 50 Mbit/s, pour différentes configurations en nombre de jetons (N_T). La limitation à 2 Mbit/s dans le mode interne (imposée par le diviseur de la sortie de l'anneau) est due au fait que le module de transfert n'avait pas été testé avec des débits hauts au moment du développement du circuit. Nous rappelons que le taux minimal d'entropie par bit estimé pour cette configuration quand le taux d'occupation est optimal est $H_m = 0.67$.

Comme 163 est un nombre premier, quelque soit N_T, l'anneau est dans une configuration multiphases (avec 163 phases équi-réparties). Par contre, la fréquence d'oscillation varie en fonction du taux d'occupation dans le mode régulier de l'anneau auto-séquencé (jusqu'à environ 50% dans nos mesures du chapitre 4). Les configurations au delà de $N_T = 88$ présentent un mode d'oscillation en rafale.

L'intégralité des tests AIS31 passe sans que nous ayons eu à appliquer un post-traitement avec un débit de 2 Mbit/s, et cela pour toutes les configurations de N_T qui présentent un mode régulier. De plus, dans ce mode interne, l'instant d'échantillonnage est fixe par rapport aux fronts de l'anneau, ce qui veut dire que le taux d'entropie par bit ne varie pas. D'ailleurs, pour la configuration $N_T = 90$, la sortie affiche une suite continue de '1'. D'autre part, pour ce débit, le *jitter* accumulé sur le signal d'échantillonnage (même s'il est issu de l'anneau lui-même) est relativement important par rapport à la résolution de phase de l'anneau auto-séquencé, ce qui augmente considérablement les taux d'entropie réels par rapport au seuil bas calculé dans le modèle (ce seuil est alors trop pessimiste).

N_T	FIPS	T0	T1	T2	T3	T4	T5	T6	T7	T8
76	100%	P	P	P	P	P	P	P	P	P
78	100%	P	P	P	P	P	P	P	P	P
80	100%	P	P	P	P	P	P	P	P	P
82	100%	P	P	P	P	P	P	P	P	P
84	100%	P	P	P	P	P	P	P	P	P
86	100%	P	P	P	P	P	P	P	P	P
88	0%	P	-	-	-	-	-	-	-	-
90	0%	-	-	-	-	-	-	-	-	-
92	0%	-	-	-	-	-	-	-	-	-

Table 6.4 – Résultats des tests AIS31 pour un STRNG de 163 étages, implanté dans une technologie CMOS 350 nm, dans son mode interne pour différentes configurations en jetons (N_T), à un débit de 2 Mbit/s

N_T	FIPS	T0	T1	T2	T3	T4	T5	T6	T7	T8
76	100%	P	P	P	P	P	P	P	P	P
78	100%	P	P	P	P	P	P	P	P	P
80	100%	P	P	P	P	P	P	P	P	P
82	99%	P	-	P	P	P	P	P	P	P
84	99%	P	-	P	P	P	P	P	P	P
86	79%	-	-	-	P	P	P	P	P	-
88	0%	-	-	-	-	-	-	-	-	-
92	0%	-	-	-	-	-	-	-	-	-

Table 6.5 – Résultats des tests AIS31 pour un STRNG de 163 étages, implanté dans une technologie CMOS 350 nm, dans son mode externe pour différentes configurations en jetons (N_T), à un débit de 50 Mbit/s

Nous avons augmenté le débit à 50 Mbit/s en utilisant l'horloge externe (mode externe du générateur). Avec ce débit, l'impact de la modification de la fréquence semble plus apparent. La fréquence optimale est obtenue autour de $N_T = 78$. La configuration avec $N_T = 82$ montre déjà quelques défauts statistiques mineurs, notamment un léger biais (T1 est un test de biais). Les configurations en mode rafale (au delà de 88 jetons) ne passent par contre aucun test. Enfin, précisons que le taux d'occupation optimal (au milieu de la gamme de jetons qui donne lieu à une propagation régulière) peut être facilement être déterminé par la simulation électrique pendant la phase de développement du générateur.

Finalement, dans cette implantation que nous avons dimensionnée *a priori* à l'aide du modèle et de la mesure sur les anneaux, les tests AIS31 ont systématiquement passés lors de l'étape d'évaluation avec un débit de 2 Mbit/s (sans que nous ayons à ajuster le nombre d'étages ni même à appliquer un post-traitement), et jusqu'à 50 Mbit/s si le nombre de jetons est optimal (et donc la fréquence de l'anneau maximale). Par ailleurs, comme le *design* est mieux maîtrisé, la sortie ne présente pas de biais allant dans un sens unique comme c'est le cas pour les implantations FPGA.

Cette implantation peut éventuellement être utilisée sans post-traitement.

4. Conclusion

Dans ce chapitre, nous avons évalué le coeur du STRNG dans des cibles FPGA, et nous avons illustré sa méthode de dimensionnement en utilisant les mesures du chapitre 4 et le modèle du chapitre 5 pour la conception d'un circuit prototype contenant le coeur du générateur dans une technologie CMOS 350 nm.

En ce qui concerne l'imprévisibilité de la sortie du générateur, les contraintes d'implantation font qu'il n'est pas toujours possible d'atteindre un taux d'entropie minimal par bit supérieur à 0.99. Dans les cibles FPGA, il faut des anneaux d'environ 1000 étages pour atteindre ce taux (600 étages dans la technologie CMOS 350 nm). Par contre des taux avoisinant 0.7 peuvent facilement être obtenus (environ 250 étages dans les FPGA et 180 étages dans la technologie CMOS 350 nm). Dans ce cas, le filtre de parité est réglé de manière à obtenir un taux avoisinant 0.99 au niveau des bits compressés. Mais comme les débits initiaux sont très importants, la perte de débit due à l'utilisation du filtre n'est pas problématique.

En ce qui concerne l'analyse statistique dans les FPGA, nous montrons que des configurations à partir d'une centaine d'étages avec un filtre de parité d'ordre 4 passent l'intégralité des tests statistiques avec des débits de l'ordre de la centaine de Mbit/s. Par contre, ces implantations semblent avoir des défauts électriques (notamment au niveau du rapport cyclique des signaux) qui introduisent un biais dans les bits de sortie. L'utilisation de post-traitement arithmétique est presque toujours nécessaire, mais celui-ci n'a pas besoin d'être important. Un filtre de parité d'ordre 4 a été généralement suffisant pour corriger la plupart des défauts dans les configurations évaluées.

Enfin, nous avons évalué des séquences issues du circuit prototype (STRNG de 163 étages en technologie CMOS 350 nm) en utilisant les procédures préconisées par AIS31 pour l'évaluation de la sortie brute d'un TRNG et de sa sortie post-traitées. Les configurations testées ont réussi tous les tests préconisés par AIS31 sans post-traitement avec un débit de 2 Mbit/s (jusqu'à 50 Mbit/s si le nombre de jetons est optimal), y compris les tests censés être appliqués après un post-traitement. Bien sûr, il n'est pas exclu de viser des débits beaucoup plus important si on augmente le nombre d'étages de l'anneau.

Conclusions et perspectives

Les générateurs de nombres véritablement aléatoires (TRNG) sont des composants importants dans certaines applications cryptographiques sensibles (génération de clés de chiffrement, génération de signatures DSA, etc). Dans de telles applications, les TRNG sont des composants très bas-niveau, leur sécurité est donc un point critique : une faille dans le TRNG peut remettre en question la sécurité de tout le système cryptographique. L'objectif de cette thèse était d'étudier les avantages des techniques de conception asynchrone pour la conception de générateurs de nombres véritablement aléatoires (TRNG) sûrs et robustes. Dans ce travail, nous nous sommes en particulier intéressés à des oscillateurs numériques appelés anneaux auto-séquencés. Ceux-ci exploitent des protocoles utilisant des requêtes et acquittement pour séquencer les données qui y circulent. En exploitant les propriétés uniques de ces anneaux, nous proposons un nouveau principe de TRNG, avec une étude théorique détaillée sur son fonctionnement, et une évaluation sur des cibles ASIC et FPGA. Ce chapitre synthétise les principaux résultats de ces travaux, présente les études complémentaires à réaliser et détaille les perspectives de recherche ouvertes par ces travaux.

Bilan et contributions

Le chapitre 1 a proposé un état de l'art sur les générateurs de nombres aléatoires avec un point de vue détaillé sur la sécurité des TRNG implantés dans les circuits numériques. La sécurité d'un TRNG est liée à deux facteurs : l'imprévisibilité de sa sortie et les moyens mis en oeuvre pour la garantir quelque soient les conditions (défaillances, perturbations externes, attaques, etc). Beaucoup de principes de TRNG existent dans la littérature, mais peu d'entre eux sont axés sur leur sécurité (l'imprévisibilité de leur sortie est rarement rigoureusement établie). Ce chapitre a donc permis d'exposer la motivation à l'origine de ces travaux.

La source d'aléa la plus pratique dans les circuits numériques est le *jitter* car il est omniprésent et facilement accessible. Les techniques qui extraient l'aléa du *jitter* nécessitent souvent de générer des signaux oscillants de manière précise. Les structures qui permettent de réaliser cela sont souvent soit peu précises (anneaux à inverseurs) soit coûteuses (par exemple une PLL). Le chapitre 2 a montré comment il est possible de générer de tels signaux de manière précise avec des structures peu coûteuses (à base de cellules logiques numériques) qui utilisent des protocoles de communication asynchrones, appelés anneaux auto-séquencés. Nous avons présenté les anneaux auto-séquencés, leurs propriétés et avantages pour générer des signaux oscillants avec une haute précision temporelle. Les principales caractéristiques de ces anneaux sont le réglage dynamique de la fréquence d'oscillation, l'adaptation à la génération de signaux multiphasés avec une très fine résolution temporelle et leurs propriétés de robustesse. Ce chapitre a donc justifié le choix de la piste asynchrone pour la conception d'un nouveau principe de TRNG sûr et robuste.

Le chapitre 3 a présenté le nouveau TRNG à base d'anneau auto-séquencé (STRNG), qui constitue la principale contribution de ce travail de thèse. Nous avons montré dans ce chapitre que les propriétés uniques des anneaux auto-séquencés offrent un cadre permettant d'appliquer un schéma d'extraction simple et compréhensible pour générer des nombres aléatoires à partir du *jitter*. Dans ce TRNG, l'entropie en sortie est directement réglée en choisissant le nombre d'étages de l'anneau auto-séquencé. Ceci permet d'adapter l'extraction d'entropie en fonction de l'amplitude du *jitter* (quelle que soit sa valeur) tout en fonctionnant à un débit optimal. D'autre part, le mode interne (utilisant une horloge d'échantillonnage issue de l'anneau auto-séquencé) offre des possibilités de *monitoring* avancées grâce au fait qu'il ne génère aucun pseudo-aléa. Le schéma proposé est suffisamment simple pour permettre une modélisation réaliste de l'entropie en sortie du générateur. Cela nécessite d'abord de bien caractériser la source d'aléa utilisée (*jitter* de l'anneau auto-séquencé).

Le chapitre 4 a présenté la première caractérisation du *jitter* dans des anneaux auto-séquencés. Nous avons montré que le *jitter* dans ces anneaux présente un profil gaussien avec une amplitude très faible. Ceci est mis en évidence par la simulation numérique, mais aussi grâce à des mesures sur des cibles FPGA (Altera Cyclone III et Xilinx Virtex 5) et ASIC (CMOS 350 nm). Nous avons montré que le *jitter* mesuré sur la période d'oscillation (*period jitter*) ne varie pas avec le nombre d'étages de l'anneau, il varie très peu avec le taux d'occupation de l'anneau (cette variation est à peine détectable dans le circuits implantés). Par ailleurs, nous avons quantifié le *period jitter* en fonction du bruit généré localement dans les cellules de bases. Les expériences réalisées suggèrent que le *period jitter* est principalement dû au bruit généré localement dans l'étage où on effectue la mesure. De plus, comme les fréquences des anneaux auto-séquencés sont généralement hautes (proches de celles d'anneaux à inverseurs de 2 ou 3 étages), les bruits de type *flicker* sont très limités. Enfin, le fait que l'amplitude du *jitter* soit très faible n'est pas problématique dans le cas du STRNG : l'extraction d'entropie peut facilement être adaptée quelque soit l'amplitude du *jitter*. Ce chapitre a donc montré que les anneaux auto-séquencés sont très adaptés comme sources d'entropie : le *jitter* dans ces anneaux présente une faible amplitude, qui est décorrélée de la configuration de l'anneau (notamment le nombre d'étages), mais aussi de la fréquence de fonctionnement (nous n'avons pas constaté de changement dans les valeurs mesurées pour des anneaux ayant plus de 200 MHz de différence de fréquence). De plus ces structures sont robustes aux fluctuations environnementales (nous avons illustré cela en analysant la variabilité par rapport à la tension d'alimentation).

Le chapitre 5 a proposé un modèle qui lie l'entropie en sortie du générateur aux caractéristiques de la source d'aléa et aux paramètres de l'extracteur. Nous avons estimé un seuil bas d'entropie par bit de sortie en fonction du nombre d'étages L de l'anneau auto-séquencé, de sa période d'oscillation T et de l'amplitude du *jitter* σ. Ce modèle est important car il permet le réglage du générateur (*i.e.* le choix de L) en fonction des paramètres mesurés sur la cible (T et σ) afin de garantir l'imprévisibilité de sa sortie (représentée par le seuil minimal d'entropie par bit). D'autre part, nous avons établi des conditions sur le pas d'échantillonnage afin de garantir des bits de sortie indépendants. Ensuite, nous avons proposé plusieurs stratégies pour exploiter ce modèle en fonction des exigences de sécurité de l'application et des contraintes de la cible d'implantation. Enfin, nous

avons posé les bases théoriques pour sécuriser le générateur face aux attaques et défaillances en contrôlant son fonctionnement directement au niveau de la source d'entropie. Un mécanisme de contrôle de la configuration de l'anneau auto-séquencé est proposé afin de détecter d'éventuelles attaques et défaillances directement sur la source d'aléa. Des pistes pour un *monitoring* en temps réel ou à la demande de l'entropie sont discutées.

Enfin, le chapitre 6 a proposé une évaluation du générateur dans des cibles ASIC et FPGA. Nous avons d'abord estimé l'entropie en sortie de plusieurs configurations à l'aide des mesures effectuées dans le chapitre 4 et du modèle proposé dans le chapitre 5 pour illustrer le principe de réglage du générateur. Ensuite, nous avons évalué les propriétés statistiques de la sortie à l'aide des batteries FIPS 140-1, AIS31 et NIST SP 800-22. Pour l'implantation FPGA, nous avons testé plusieurs configurations d'anneaux auto-séquencés à des débits de 16 Mbit/s et 400 Mbit/s. Bien que les résultats statistiques soient corrects, il semble que les performances soient moins bonnes que dans le circuit ASIC. En particulier, nous avons constaté la présence d'un biais qui semble lié à des problèmes électriques dans l'implantation (notamment le rapport cyclique des signaux). Cependant, ces défauts statistiques semblent être faciles à corriger. Par exemple, pour des configurations avec un taux d'entropie théorique proche de 0.90, il suffit d'un filtre de parité d'ordre 2 pour éliminer ces défauts (ce qui diminue le débit par deux). En général, les configurations autour d'une centaine d'étages avec un filtre d'ordre 4 passent l'ensemble des tests avec un débit de l'ordre de la centaine de Mbit/s. Pour l'implantation ASIC, nous avons exploité les stratégies basées sur la mesure proposées dans le chapitre 5 pour faire un dimensionnement complètement *a priori*. A l'aide des mesures sur les anneaux, nous avons choisi le nombre d'étages (163) de l'anneau pour obtenir un taux minimal d'entropie en sortie proche de 0.67. Le STRNG implanté a fourni des séquences qui passent tous les tests FIPS et AIS31 avec un débit de 2 Mbit/s sans aucun post-traitement (et jusqu'à 50 Mbit/s si le taux d'occupation de l'anneau est optimal). Ce chapitre a donc montré comment il est possible d'exploiter le modèle et les mesures pour faire un dimensionnement du TRNG *a priori* : si le nombre d'étages est choisi conformément par rapport à la mesure, alors la sortie est imprévisible (ceci est vérifié grâce au taux d'entropie calculé) et a de bonnes propriétés statistiques (ceci est vérifié à l'aide des batteries de tests standard).

Finalement, au cours de ce travail, nous avons proposé un nouveau principe de TRNG utilisant des anneaux auto-séquencés (qui sont des oscillateurs utilisant les techniques de conception asynchrones). Ce TRNG apporte deux avantages significatifs par rapport aux techniques d'extraction du *jitter* proposées dans la littérature : 1) un contrôle très précis de l'entropie en sortie à l'aide du nombre d'étages de l'anneau auto-séquencé (le *design* est donc facilement adaptable en fonction de la cible d'implantation et du niveau de sécurité visé), 2) un mode de fonctionnement qui ne présente pas de pseudo-aléa, ce qui permet un *monitoring* simple et efficace de l'entropie réelle en sortie. Nous avons proposé une méthodologie complète pour caractériser ce générateur, le dimensionner et le sécuriser face aux attaques et défaillances. Plusieurs implantations dans des cibles FPGA ont été réalisées, ainsi que deux circuits intégrés ASIC en technologie CMOS 350 nm. Ce TRNG a été évalué de la manière suivante : 1) la source d'aléa a été caractérisée et quantifiée 2) le mécanisme d'extraction d'entropie a été modélisé 3) l'entropie en sortie du générateur a été estimée en fonction des paramètres de l'extracteur (réglables) et des mesures sur la source d'aléa

4) la qualité statistique des bits de sortie a été validée à l'aide des batteries de tests standard.

Les débits permis par ce générateur sont très hauts et proches des limites permises par les technologies d'implantation. Bien que l'implantation nécessite une étape de placement/routage mais aussi des mesures sur les anneaux (si on veut correctement dimensionner le générateur), ceci n'est pas problématique dans l'optique d'une utilisation industrielle car cette étape est à effectuer une fois pendant la phase de développement pour une technologie donnée (et non pour chaque implantation). D'autre part, l'adaptabilité du *design* (en jouant sur un compromis débit/taille à l'aide de l'utilisation du filtre de parité) fait qu'il est presque toujours possible de réduire la taille et la consommation quitte à réduire le débit (sachant que le débit initial est déjà très haut). Enfin, en ce qui concerne l'évolution des technologies (*technology shrink*), il est avéré que les anneaux auto-séquencés supportent bien la diminution des dimensions des transistors (il existe des implantations d'anneaux auto-séquencés qui ont été validées dans les technologies les plus fines, jusqu'à 28 nm). D'autre part, on peut supposer que la valeur relative du *jitter* par rapport à la période d'oscillation augmente (σ/T augmente). Cela veut dire que pour une configuration donnée, l'entropie devrait augmenter quand on passe à une technologie plus fine.

Travaux complémentaires

Dans le chapitre 4, nous avons effectué des mesures de *jitter* dans des cibles ASIC et FPGA. Comme la mesure est externe, nous avons émis le doute que les valeurs mesurées soient plus grandes que celles à l'intérieur du circuit puisque la circuiterie d'entrée/sortie, la connectique et la sonde peuvent également introduire du bruit. Comme les valeurs qu'on cherche à mesurer sont très faibles, il n'est pas exclu qu'elles soient du même ordre que l'incertitude introduite par l'appareillage de mesure. Dans ce cas, il est plus judicieux d'effectuer ces mesures à l'intérieur du circuit : ceci nécessite d'imaginer des moyens efficaces pour réaliser cette mesure. Une piste est proposée dans le chapitre 5 (utilisant la résolution de phase d'un anneau auto-séquencé comme résolution temporelle de la mesure) mais sa réalisation et son modèle théorique présentent encore des défauts. Il faudrait étudier cette piste de près et résoudre les problèmes d'implantation dus aux ressources limitées dans les FPGA (arbre d'horloge pour la gestion du *skew*, routage des cellules ...). Sinon, pour les implantations ASIC, il serait aussi intéressant d'envisager des mesures sous pointe dans un environnement dédié.

Dans la première partie du chapitre 5, nous avons modélisé l'entropie en sortie du générateur en se basant uniquement sur l'imprévisibilité du *jitter* de l'anneau auto-séquencé. On pourrait améliorer ce modèle en prenant en compte le *jitter* de l'horloge d'échantillonnage : le seuil bas d'entropie qu'on a calculé sera alors moins pessimiste, notamment quand l'horloge d'échantillonnage a une fréquence basse. D'autre part, nous avons analysé les dépendances entre les positions des fronts d'un anneau auto-séquencé afin de déduire une fréquence d'échantillonnage permettant d'avoir des bits non corrélés à la sortie du générateur. Cet aspect doit être quantifié mathématiquement en calculant l'entropie conditionnelle d'un bit en connaissant ses prédécesseurs, en fonction du pas d'échantillonnage. Ceci doit être fait en utilisant l'équation *Charlie* qui définit le lien entre les positions des fronts dans l'anneau.

Dans la deuxième partie du chapitre 5, nous avons analysé un modèle de menace du générateur qui présente quelques attaques éventuelles sur le générateur. Ces attaques doivent être validées afin de vérifier si elles sont efficaces pour le STRNG ou non, mais aussi pour tester l'efficacité des tests embarqués et alarmes, qui doivent d'ailleurs être implantés dans le matériel.

Perspectives de recherche

Les anneaux auto-séquencés ont une fréquence qui est facilement reconfigurable de manière dynamique. De plus, les travaux présentés dans le chapitre 4 ont mis en évidence le fait que le *jitter* dans ces anneaux est très faible, et que ces oscillateurs sont plus robustes aux variations de tension d'alimentation que les anneaux à inverseurs. Ceci fait de ces oscillateurs de très bon candidats pour la génération de signaux d'horloges précis dans le matériel (en effet les facteurs les plus importants sont un faible bruit, une bonne robustesse et une bonne précision du réglage de fréquence). Les caractéristiques de faible bruit sont particulièrement intéressantes dans les application RF (radio-fréquences) où les signaux manipulés ont une très faible puissance.

Naturellement, une première idée qui nous vient à l'esprit comme perspective de travaux sur ce TRNG est d'exploiter la même structure pour construire une fonction physique non clonable (*Physical Unclonable Function*). Si dans le TRNG, l'idée est de régler la résolution de phase de l'anneau de manière à capter les variations dues au bruit, alors on peut très bien imaginer qu'on puisse régler cette résolution de phase dans une valeur intermédiaire qui est suffisamment fine pour capter des variations dues aux procédés de fabrications (entre des puces différentes), mais suffisamment grande pour ne pas être influencée par le bruit (à l'intérieur d'une même puce). De plus, si on effectue ce réglage uniquement avec le nombre d'événements de l'anneau (sans changer le nombre d'étages), alors on peut construire une structure duale qui fournit à la fois des nombres aléatoires, mais aussi une signature digitale non-clonable du circuit pour fournir des services d'authentification et d'identification (utilisés dans les protocoles cryptographiques mais aussi pour la lutte contre la contre-façon, l'activation en ligne de produits, etc).

De manière plus générale, le principe de TRNG à base d'anneau auto-séquencé présenté dans ce travail de thèse laisse entrevoir des possibilités beaucoup plus vastes exploitant la propriété de finesse des phases des signaux de l'anneau. De notre point de vue, c'est la propriété qui ouvre le plus de perspectives de recherche, la plus importante et la plus vaste étant la mesure de temps. Le chapitre 5 a posé les bases théorique d'une mesure de temps exploitant la résolution de phase d'un anneau auto-séquencé comme résolution de mesure (nous rappelons que celle-ci peut être réglée aussi fine que voulu à l'aide des paramètres de l'anneau). Les perspectives de recherche exploitant cette piste sont nombreuses, et les domaines d'application très variés : appareillage et outils de mesure (*e.g.* un oscilloscope), traitement de signaux de radio impulsionnelle UWB (*Ultra Wide Band*), géolocalisation à haute précision, PUF basés sur la mesure de délais tels que les PUF à base d'anneaux à inverseurs et les PUF utilisant un arbitre (en améliorant la précision de la mesure de temps dans ces techniques) ...

Publications et communications de l'auteur

Brevet

L. Fesquet, J. Hamon et A. Cherkaoui. *Générateurs de nombres aléatoires vrais*. Brevet déposé le 6 Février 2012 à l'INPI (Institut National de la Propriété Intellectuelle), N° FR 12 51079.

Conférences internationales avec comité de lecture et acte

[CFAF12] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. *Comparison of self-timed ring and inverter ring oscillators as entropy sources in FPGAs*. In Design, Automation Test in Europe Conference Exhibition (DATE), 2012, pages 1325–1330, March 2012.
doi : 10.1109/DATE.2012.6176697

[CFAF13] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. *A self-timed ring based true random number generator*. In Asynchronous Circuits and Systems (ASYNC), 2013 IEEE 19th International Symposium on, pages 99–106, May 2013.
doi : 10.1109/ASYNC.2013.15

[CFFA13] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert. *A very high speed true random number generator with entropy assessment*. In Guido Bertoni and Jean-Sébastien Coron, editors, Cryptographic Hardware and Embedded Systems - CHES 2013, volume 8086 of Lecture Notes in Computer Science, pages 179–196. Springer Berlin Heidelberg, 2013.
doi : 10.1007/978-3-642-40349-1_11

Workshops

A. Cherkaoui, V. Fischer, L. Fesquet et A. Aubert. *A new robust true random number generator using self-timed rings*. International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi'2011), Bochum, Allemagne, 2011.

A. Cherkaoui, V. Fischer, L. Fesquet et A. Aubert. *Asynchronous self-timed rings for randomness generation*. International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi'2012), Saint-Etienne Goutelas, France, 2012.

Communications sans acte

A. Cherkaoui, V. Fischer, L. Fesquet et A. Aubert. *Self-timed rings as entropy sources in FPGA*. Communication orale, International Workshop on Practical Hardware Innovations in Security Implementation and Characterization (PHYSICS'2011), Gardannes, France, 2011.

A. Cherkaoui, V. Fischer, L. Fesquet et A. Aubert. *Self-timed rings as entropy sources in FPGA*. Communication orale, journées du projet SEmba, Valence, France, 2011.

A. Cherkaoui, V. Fischer, L. Fesquet et A. Aubert. *Self-timed rings as sources of entropy*. Communication écrite et orale, journées du GDR SoC-SiP, Paris, France, 2012.

A. Cherkaoui, V. Fischer, L. Fesquet et A. Aubert. *Génération de nombres aléatoires à l'aide de circuits asynchrones*. Communication orale, journées du projet SEmba, Saint Germain au Mont d'Or, France, 2013.

A. Cherkaoui et V. Fischer. *True random number generators : security aspects and new design using asynchronous techniques*. Sécurité des systèmes embarqués, séminaire organisé par l'IRISA et le DGA-MI, Rennes, France, 2014.

Posters

A. Cherkaoui, V. Fischer, L. Fesquet et A. Aubert. *A new true random numbers generator using self-timed rings*, Journée de la recherche de l'école doctorale SIS 488, Saint-Etienne, France, 2011.

L. Fesquet, A. Cherkaoui, V. Fischer et A. Aubert. *Self-timed rings as entropy sources*. IEEE 18th International Symposium on Asynchronous Circuits and Systems (ASYNC), Lyngby, Danmark, 2012.

Appendices

Annexe A

Principes de TRNG dans les circuits numériques

Cette annexe est un complément du chapitre 1. Elle présente quelques principes de TRNG proposés dans la littérature. Nous décrivons essentiellement les *designs* qui ne sont pas discutés dans le chapitre 1, sans nous intéresser à leur sécurité.

1. TRNG proposé par Bagini *et al.*

Résumé de la technique Une source de bruit est amplifiée et transformée en signal numérique via un comparateur, celui-ci est échantillonné pour générer les nombres aléatoires.

Le TRNG présenté par Bagini *et al.* dans [BB99] utilise le bruit thermique d'un composant analogique externe comme source d'entropie. Le schéma du TRNG est présenté dans Fig. 1.1. Le signal bruité est d'abord filtré, amplifié, et ensuite transformé en un signal numérique via le comparateur à hystérésis. Les auteurs proposent une étude théorique assez complète qui permet d'exprimer la corrélation des bits de sortie en fonction des paramètres internes du générateur, et ainsi établir les débits maximaux pour obtenir une indépendance des bits de sortie. Néanmoins, les auteurs ne donnent aucun résultat concret d'implantation.

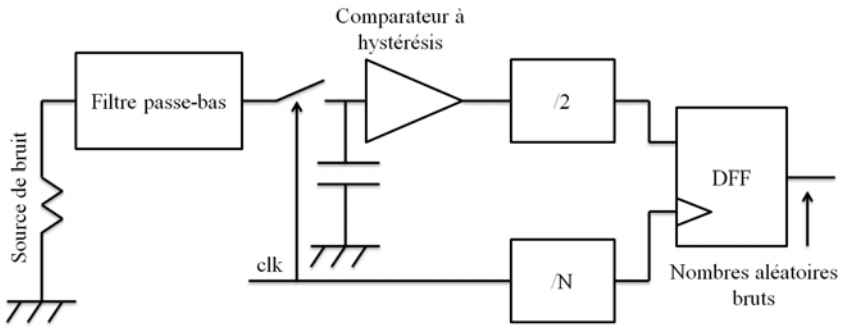


Figure 1.1 – Schéma de principe du TRNG proposé dans [BB99]

2. TRNG proposé par Jun *et al.*

Résumé de la technique La différence entre deux sources de bruit est amplifiée et alimente un VCO, celui-ci est échantillonné pour générer les nombres aléatoires

Ce principe de TRNG, très proche de celui présenté dans [BB99], a été développé par *Intel*, et intégré à une de ses puces pour pallier au cruel manque de vraie source d'aléa dans les systèmes d'exploitation. L'évaluation de ce générateur par la société *Cryptography Research* est présentée dans [JK99].

Dans cette architecture, présentée dans Fig. 1.2, deux sources de bruit (résistances intégrées) sont utilisées. Leurs signaux de sortie sont comparés, le résultat est ensuite amplifié et appliqué en entrée d'un oscillateur contrôlé en tension (*Voltage Controlled Oscillator - VCO*). Les nombres aléatoires bruts sont obtenus en échantillonnant la sortie sortie du VCO à intervalles réguliers. La structure différentielle utilisant deux sources de bruit indépendantes permet de rendre l'architecture plus robuste aux perturbations externes et fluctuations environnementales. Les bits de sortie sont traités via un post-traitement arithmétique de type décorrélateur de Von-Neumann. Avant d'être mises à disposition de l'usager, un post-traitement cryptographique est appliqué aux suites binaires générées. Celui-ci consiste en une fonction de hachage SHA-1.

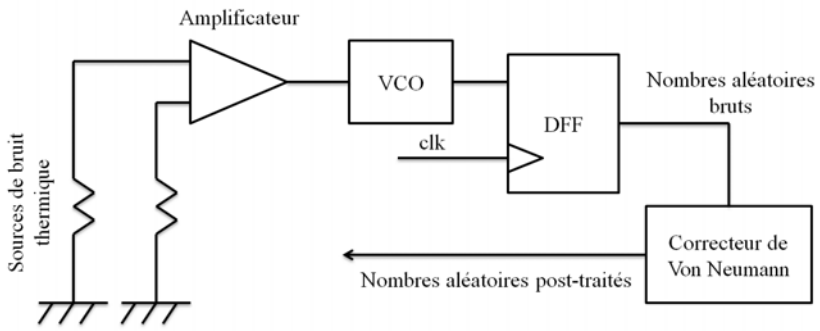


Figure 1.2 – Architecture du TRNG développé par *Intel* dans la fin des années 90 ([JK99])

Les auteurs évaluent des séquences de plus de 80 Mb via les tests NIST, FIPS 140-1 et DIEHARD. Celles-ci sont issues à la fois de la sortie du coeur du générateur (nombres aléatoires bruts) et de la sortie du post-traitement. Bien que les résultats ne soient pas donnés dans le papier, les auteurs laissent entendre que les tests passent avec succès sur les données post-traitées mais que les données brutes présentent un biais important. Les auteurs ne discutent pas de la modélisation du générateur mais on peut supposer que celle-ci est très proche du travail présenté dans [BB99]. Enfin, les débit atteints avoisinent les 75 Kb/s.

3. TRNG proposé par Killman *et al.*

Résumé de la technique La différence entre deux sources de bruits alimente un trigger de Schmitt, les nombres aléatoires sont extraits en mesurant les durées des niveaux hauts de ce signal

Dans ce générateur, les signaux bruités sont issus de deux diodes zener. La différence entre les deux tensions (théoriquement nulle en moyenne) est amplifiée puis vient alimenter un trigger de Schmitt (c.f. Fig. 1.3). Celui-ci produit alors, en fonction du bruit dans les deux diodes, une série de '1' et de '0' dont la durée est aléatoire. Chaque transition '0' vers '1' déclenche une première bascule qui inverse alors sa sortie (*toggle flip-flop*). Une deuxième bascule échantillonne ce signal tous les n cycles d'horloge. Le nombre (en base 2) de transitions '0' ver '1' durant n cycles d'horloge donne alors le nombre aléatoire brut r_n (issu du signal analogue numérisé). Le post-traitement arithmétique est donné par la formule $y_{n+1} = y_n \oplus r_n$, où y_n est le $n^{ième}$ nombre post-traité (i.e. nombre aléatoire interne).

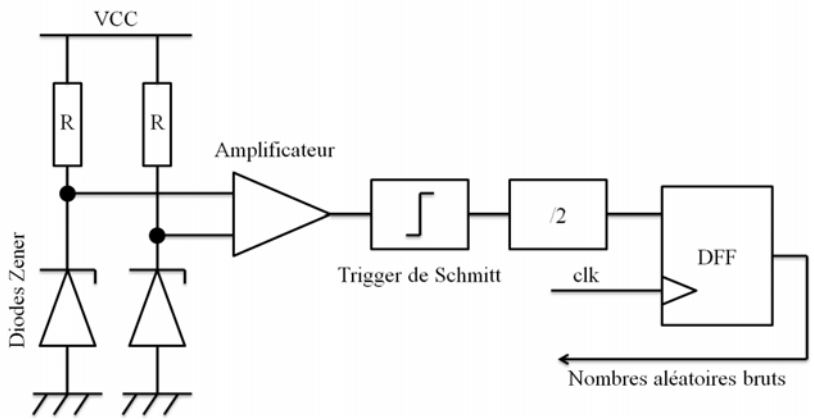


Figure 1.3 – Architecture du TRNG proposé dans [KS08]

La partie la plus intéressante de cette article concerne la méthode d'évaluation du TRNG qui est en partie conforme aux recommandations d'AIS31. Les auteurs proposent un modèle stochastique grâce auquel ils analysent le comportement du TRNG et expriment un estimateur robuste pour l'entropie en sortie du générateur en fonction de ses paramètres internes. Ensuite, les auteurs proposent un test en ligne grâce auquel ils valident son bon fonctionnement. Le débit atteint par leur design est de 500 Kb/s.

4. TRNG proposé par Danger *et al.*

Résumé de la technique Forcer une bascule dans un état de métastabilité pour produire des nombres aléatoires

Le principe du TRNG proposé dans [DGH07], illustré dans Fig. 1.4, est d'utiliser un même signal comme entrée d'horloge et de donnée d'une bascule *flip-flop*. En ajustant le délai d_1 par rapport au délai d_2, la donnée peut être échantillonnée à la limite du point de basculement de la porte CMOS (i.e. autour de $V_{dd}/2$). Le bruit thermique et les conditions environnementales déterminent alors si la tension bascule vers le niveau haut, le niveau bas, ou un état métastable qui se résout en une durée τ dépendant également du bruit et conditions environnementales. Cette

structure permet en théorie des débits très importants mais elle est très sensible aux conditions environnementales (le calibrage des délais peut être perdu avec la moindre fluctuation de tension et température).

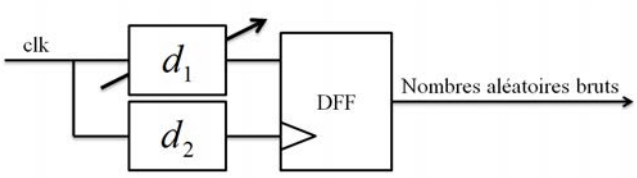


Figure 1.4 – Schéma de principe de la méthode utilisée dans [DGH07]

Pour pallier au problème posé par les fluctuations externes, les auteurs proposent une architecture composée d'une chaine de délais et de bascules comme cela est illustré dans Fig. 1.5. Le signal traverse n délais et vient à chaque fois alimenter une nouvelle bascule, les sorties de ces bascules sont combinées avec une fonction OU exclusif (XOR). Ainsi, plus le nombre de bascules est important, et plus la résolution de délai est fine, plus il est probable qu'une des bascules rentre dans un état métastable et produise un bit aléatoire. Le bit aléatoire est transmis en sortie du générateur via l'arbre de XOR.

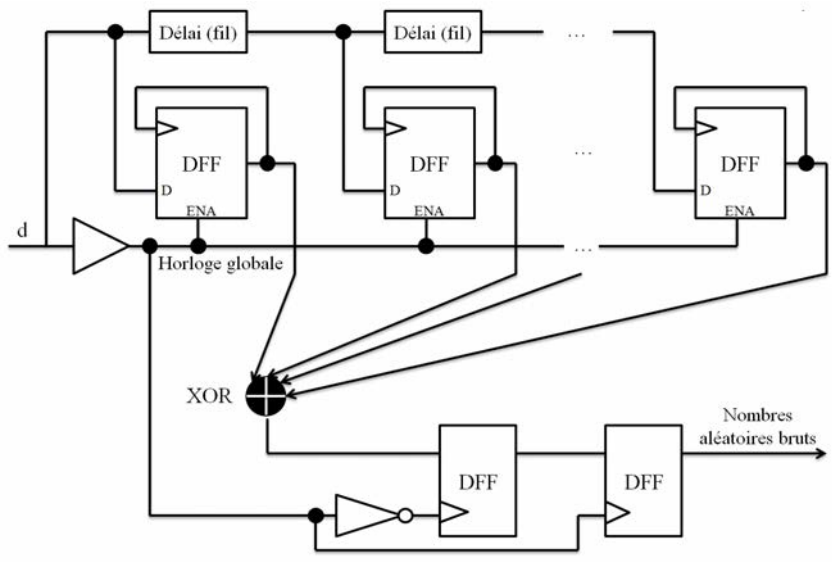


Figure 1.5 – Architecture du TRNG proposé dans [DGH07]

Dans une version améliorée, proposée dans [DGH09], les auteurs proposent de contrôler de manière dynamique les délais en fonction d'un test de biais intégré. Les suites générées passent les tests NIST SP 800-22 avec un débit assez important de 20 Mb/s après avoir été post-traitées par un correcteur de Von-Neumann.

5. TRNG proposé par Vasyltsov *et al.*

Résumé de la technique Provoquer l'état de métastabilité de cellules inverseuses pour initialiser de manière aléatoire un anneau à inverseurs

Le TRNG proposé dans [VHKK08] tire parti de l'état de métastabilité de cellules inverseuses pour initialiser de manière aléatoire un anneau à inverseurs. Ce principe reprend en partie l'idée du TRNG proposé dans [EHK+03], mais l'implantation proposée est décrite de manière plus détaillée. L'architecture proposée, nommée META-RO, est décrite dans Fig. 1.6.

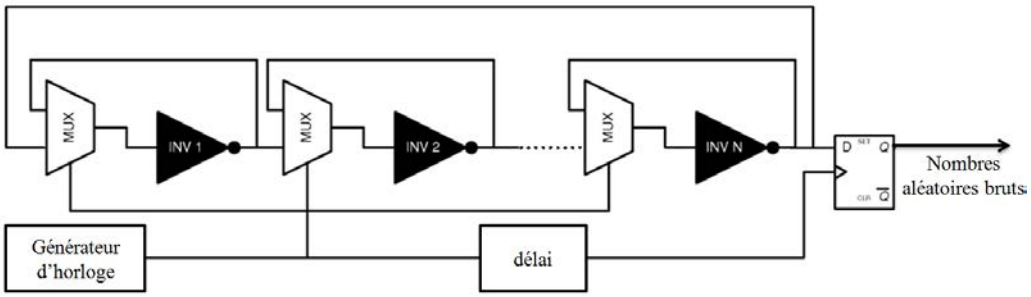


Figure 1.6 – Architecture du TRNG proposé dans [VHKK08]

Cette structure fonctionne en deux phases séquencées par une horloge *clk*, elles sont illustrées dans Fig. 1.7. Dans la première phase, les étages de l'anneau sont re-bouclés sur eux même : selon les auteurs, chaque inverseur entre alors dans un état de métastabilité avec à sa sortie une tension proche de $V_{dd}/2$), et peut être considéré comme une source indépendante de bruit. Lors de la seconde phase, les inverseurs sont reconnectés sous forme d'anneau. Les valeurs d'initialisation des étages sont à ce moment intimement liées aux fluctuations de bruit dans chaque inverseur. Les oscillations en régime transitoire de l'anneau (s'il elles ont lieu) ont également une forme et durée dépendant des valeurs d'initialisation qui sont aléatoires. Ce signal est alors échantillonné à la fin de chaque cycle pour produire un nombre aléatoire brut (1 bit).

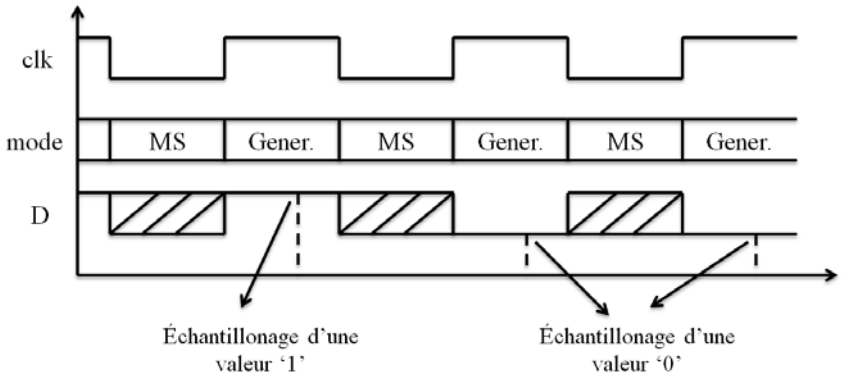


Figure 1.7 – Chronogramme de fonctionnement du TRNG proposé dans [VHKK08]

Les auteurs proposent d'évaluer le TRNG dans des cibles ASIC et FPGA. Les suites post-traitées avec un correcteur de Von Neumann passent les tests FIPS 140-2 et AIS31. Le débit atteint est de 140 Mb/s en technologie CMOS 65nm dans des conditions optimales, et entre 35 et 50 Mb/s dans des conditions non optimales de fonctionnement. Malheureusement, peu de détails sont donnés sur l'implantation FPGA.

De notre point de vue, l'obtention de l'état métastable est difficile dans les FPGA, car cela nécessite que le rebouclage d'une cellule inverseuse aie un délai de propagation très petit par rapport à celui de la cellule. Or ce délai comprend à la fois celui de l'interconnexion et du multiplexeur et peut donc être assez long en fonction des ressources disponibles dans le FPGA (par exemple la disponibilité de multiplexeur pré-connecté aux LUT). Dans ce cas, la cellule oscille comme un anneau à inverseurs et n'atteint pas l'état de métastabilité.

6. TRNG proposé par la *RAND corp.*

Résumé de la technique Un générateur d'impulsions séquence une roue électronique qui est échantillonnée une fois par seconde (accumulation du *jitter*)

RAND corp. est une institution américaine non-lucrative ayant pour but d'améliorer la politique et le processus décisionnel par la recherche et l'analyse, fondée en 1945 par l'US Air Force. Cette organisation publia dans les années 50 un ouvrage contenant un million de nombres aléatoires générés en utilisant les fluctuations temporelles d'un signal électronique [Cor66].

Le principe utilise des composants électroniques de l'époque : une source de fréquence impulsionnelle produisant 100000 pulsation par seconde en moyenne alimente une roulette électronique (compteur 5-bit) via un circuit de standardisation d'impulsions. La roue électronique effectue 3000 révolution par cycle et produit un nombre aléatoire à chaque cycle. L'aléa vient de l'accumulation des fluctuations temporelles du signal issu de la source impulsionnelle.

Les suites générées ont été validées à l'aide de trois tests statistiques : *frequency test* (test de biais), *poker test*, *serial et runs tests*. On peut remarquer que ces trois tests sont présents dans la norme FIPS actuelle. Le débit est de l'ordre d'un nombre aléatoire par seconde.

7. TRNG proposé par Fairfield *et al.*

Résumé de la technique Un oscillateur basse-fréquence échantillonne un oscillateur haute-fréquence (accumulation du *jitter*)

L'article référencé [FMC85], est fondateur dans le domaine de la génération d'aléa dans les circuits numériques. Il présente la première implantation de TRNG dans un circuit électronique intégré, ainsi qu'une analyse mathématique très complète sur son fonctionnement. La technique proposée exploite la propriété d'accumulation du *jitter* dans le temps. L'architecture utilise une simple bascule D (*flip-flop*) qui échantillonne un signal *jitté* pour en extraire les nombres aléatoires. Ce sera dorénavant un des principes de base largement repris et améliorés par la communauté scientifique ultérieurement. Le principe de fonctionnement est décrit dans Fig. 1.8. Un signal haute

fréquence, généré grâce à un oscillateur embarqué de 8 MHz, est échantillonné par un signal basse fréquence ajustable grâce à des composants externes. La valeur échantillonnée dépend des variations temporelles accumulées sur le signal haute fréquence, ainsi que du *jitter* du signal basse fréquence.

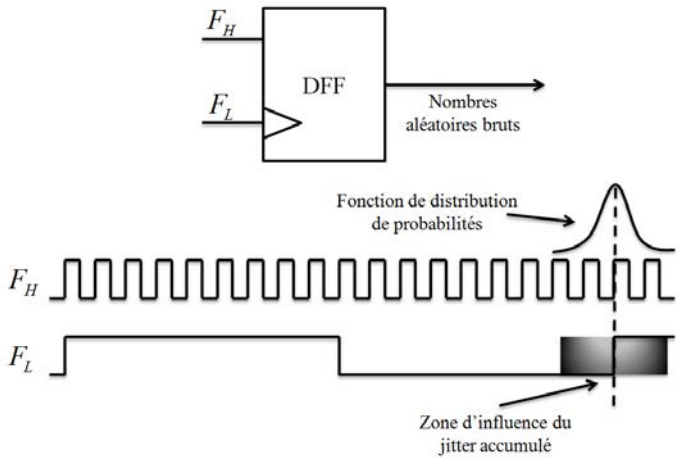


Figure 1.8 – Principe de fonctionnement du TRNG proposé dans [FMC85]

Les auteurs proposent ensuite un modèle mathématique permettant de calculer la probabilité de connaitre un bit en ayant la connaissance de son précédent, et lient cette probabilité aux paramètres du *jitter*. Ce dernier est modélisé par une loi normale et quantifié par son écart-type. Les auteurs étudient également l'effet du rapport entre les deux fréquences sur l'apparition de motifs (*patterns*) dans la suite de séquences générées. Une modélisation stochastique plus complète de ce type de générateurs est présentée dans [BLMT11].

Une analyse mathématique approfondie des suites générées est proposée et révèle que l'imperfection du rapport cyclique des signaux périodiques introduit nécessairement un biais dans les nombres aléatoires générés. Ceci est d'ailleurs un constat général pour tous les TRNG utilisant des signaux périodiques. Pour remédier à cela, les auteurs utilisent un filtre de parité qui corrige le biais au détriment d'un plus faible débit. Une autre solution proposée, plus coûteuse, consiste à utiliser un brouilleur (*scrambler*) qui réduit également les corrélation entre les bits. Les débits donnant de bonnes suites aléatoires sont de l'ordre du Kb par seconde.

18 ans plus tard, Tsoi*et al.* proposent une implantation de ce principe dans des cibles FPGA [TLL03]. Dans cette première version, le signal basse fréquence est généré à l'extérieur du FPGA, ce qui peut poser un problème de sécurité (un attaquant peut en prendre le contrôle). Pour remédier à cela, les auteurs proposent dans [TLL07] d'utiliser un oscillateur en anneau à inverseurs et un diviseur de fréquences à la place de l'oscillateur externe. D'après les auteurs, les données post-traitées par un filtre de parité passent les tests statistiques NIST SP 800-22 et DIEHARD avec un débit de 29 Kb/s.

Une autre variante de ce TRNG, implantée dans les familles FPGA de Xilinx, est proposée par Kwork *et al.* dans [KL06]. Elle utilise la DFS (*Digital Frequency synthetiser*) intégrée dans le

FPGA pour générer le signal haute fréquence. D'après les auteurs, en choisissant judicieusement les paramètres de multiplication ($m = 31$) et de division ($d = 32$), il est possible de maximiser le *jitter*. Cependant, la nature du *jitter* n'est pas discutée dans le papier. Dans [Val10], il est démontré que le *jitter* généré de cette manière est majoritairement déterministe. Le fait que les auteurs utilisent une fonction de hachage MD5 complexe et couteuse comme post-traitement semble aller dans le sens de ce constat.

Dans [BGL+03], Bucci *et al.* proposent une amélioration de cette famille de TRNG en ajoutant un réglage dynamique de la fréquence d'échantillonnage. L'objectif de ce réglage est de calibrer les fréquences des oscillateurs pour minimiser les corrélations et augmenter l'entropie en sortie du générateur. De plus, le signal basse fréquence est généré avec un oscillateur dont les caractéristiques de bruit ont été boostées en amplifiant une source de bruit. Cependant, cette source de bruit et la nature du *jitter* obtenu dans l'oscillateur ne sont pas caractérisés dans l'article. Les séquences générées passent les tests FIPS 140-2 sans post-traitement avec un débit de 10 Mbit/s (en technologie CMOS 180 nm). Ce générateur est exclusivement destiné à une implantation ASIC.

8. TRNG proposé par Bucci *et al.*

Résumé de la technique Détection du *jitter* relatif de deux oscillateurs ayant les mêmes caractéristiques

Dans [BGL+06], Bucci *et al.* présentent une nouvelle technique d'extraction du *jitter* qui est d'après les auteurs, sans mémoire (*stateless*), i.e. qu'elle produit des bits décorrélés. Le principe se base sur la détection de la course relative de deux oscillateurs ayant les même caractéristiques (comme le montre Fig. 1.9) qui sont réinitialisés à chaque fois qu'un bit aléatoire est généré. Les deux signaux oscillants sont générés à l'aide d'une chaine de délais et d'un commutateur qui permet aux signaux d'emprunter les mêmes branches afin de garantir deux fréquences égales. Le circuit de détection déclenche l'échantillonnage lorsque le *jitter* relatif accumulé des deux oscillateurs dépasse un certain seuil.

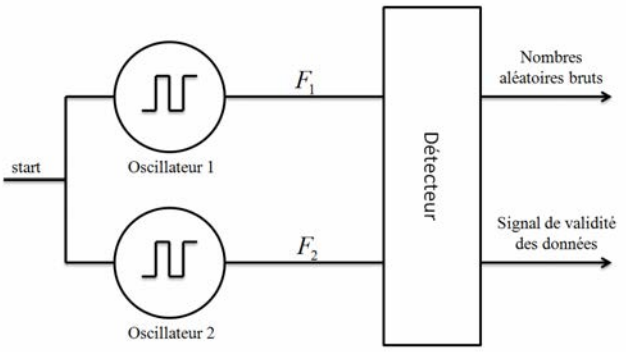


Figure 1.9 – Schéma équivalent du TRNG proposé dans [BGL+06]

Des simulations en technologie CMOS 120 nm ont permis de valider le fonctionnement général

du circuit, les auteurs prévoient un débit autour de 13 Mb/s. Cependant, aucune évaluation du principe n'est donnée dans le papier.

9. TRNG proposé par Kohlebrenner *et al.*

Résumé de la technique Échantillonnage cohérent entre deux oscillateurs ayant des fréquences très proches

Cette architecture, proposée dans [KG04] reprend le même principe que celui de [FD02] mais offre une alternative à l'utilisation de la PLL. Kohlebrenner *et al.* proposent d'utiliser deux oscillateurs ayant les mêmes caractéristiques afin de générer deux fréquences extrêmement proches. La résolution de balayage correspond alors à la différence entre les demi-périodes de ces deux oscillateurs. L'architecture proposée est décrite dans Fig. 1.10. Le signal s, de fréquence F_s, est échantillonné par le signal clk, de fréquence F_{clk} (avec $F_s \simeq F_{clk}$). Le signal résultant est composé de longues suites de '1' suivi de longues suites de '0' avec de courtes suites aléatoires intercalés entre elles. L'aléa est capté en comptant ces bit durant une fenêtre de comptage donnée par le contrôleur, celle-ci doit être réglée en fonction de la différence entre les périodes des oscillateurs.

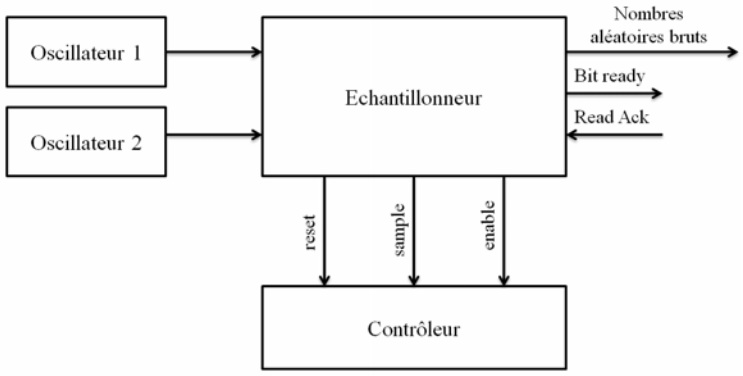


FIGURE 1.10 – Architecture du TRNG proposé dans [KG04]

Pour que la résolution de balayage soit la plus fine possible, les deux oscillateurs doivent avoir des fréquences extrêmement proches. Il s'agit de la principale difficulté concernant l'implantation de ce TRNG, en particulier dans les circuits FPGA. Il est généralement, difficile de générer des fréquences proches avec des éléments re-programmables à cause des différences de délais dues au placement/routage et variabilité des procédés de fabrications. Dans [KG04], les auteurs proposent de réaliser ces oscillateurs dans un FPGA Xilinx Virtex au sein d'un seul CLB (*Configurable Logic Block*). D'après les auteurs, la variabilité de fréquence obtenue dans ce cas est de l'ordre de 7%. Les auteurs utilisent en sortie du générateur un post-traitement de type filtre de parité pour éliminer le biais statistique. Les suites générées ont été validées à l'aide de NIST SP 800-22 avec un débit de 295 Kb/s.

Ce type de design est moins couteux d'un point de vue matériel que celui présenté dans

[FD02] (il ne nécessite pas de composant dédié), mais ne permet pas un contrôle aussi précis de la résolution de balayage. Par ailleurs, la mise en place de ce générateur n'est pas aisée car elle nécessite une phase de dimensionnement longue et impossible à automatiser. En effet, celle-ci est spécifique à chaque implantation (y compris au sein de la même famille/technologie). En particulier dans les FPGA, pour chaque carte, il faut tester différentes configuration de placement/routage jusqu'à obtenir une différence de fréquence satisfaisante. Évidemment, cela limite une potentielle utilisation industrielle tant que ce problème ne sera pas résolu.

Dans [VFA09], B. Valtchanov *et al.* proposent plusieurs implantations de ce TRNG en introduisant quelques modifications mineures (notamment l'utilisation de plusieurs instances du TRNG) et en essayant différentes sources d'horloges (oscillateurs en anneau à inverseurs, PLL et DFS) sur des cibles Actel (actuellement Microsemi) et Altera. Deux configurations utilisant des anneaux à inverseurs ont fourni des suites validées avec NIST SP 800-22 avec un débit de 2 Mb/s, elles ont néanmoins nécessité un placement/routage manuel. Par contre la configuration utilisant la PLL donne des performances similaires mais ne nécessite aucune intervention manuelle.

10. TRNG proposé par Varchola *et al.*

Résumé de la technique Capter le nombre d'oscillations en régime transitoire dans un circuit bi-stable (ce nombre est directement influencé par le *jitter*)

Dans [VD10], M. Varchola et M. Drutarobsky présentent un TRNG qui extrait le *jitter* dans un circuit bi-stable. Lorsqu'un circuit bi-stable est initialisé dans une situation conflictuelle, il oscille pendant un régime transitoire avant d'atteindre un état final stable. Les auteurs remarquent un fait intéressant : le nombre d'oscillations durant le régime transitoire est directement influencé par le *jitter*. L'architecture d'une cellule de base de ce TRNG, appelée cellule TERO (*Transient Effect Ring Oscillator*) est décrite dans Fig. 1.11. Lorsque le signal de contrôle est à '0', le schéma équivalent du circuit est un bi-stable bloqué dans un état conflictuel. Lorsque le signal de contrôle passe à '1', le circuit bi-stable oscille durant un régime transitoire, des bascules de type *toggle* sont placées à la sortie de chaque étage pour compter le nombre d'oscillations en base 2. La fréquence de fonctionnement est calibrée en fonction de la durée maximale des oscillations transitoires. Enfin, l'architecture complète du TRNG comporte une ou plusieurs instanciations de cellules TERO dont les sorties sont combinées avec un XOR pour fournir les nombres aléatoires bruts.

On peut remarquer que la structure et la présence de deux phases de fonctionnement sont très similaires au TRNG présenté dans [EHK$^+$03]. La différence entre ces deux principes tient dans le fait que le TERO-TRNG ne cherche pas à provoquer d'état de métastabilité aux niveau des étages du circuit bi-stable (la tension de sortie n'est pas dans une valeur intermédiaire au moment du démarrage), il exploite uniquement le *jitter* dans les oscillations transitoires.

Par ailleurs, l'approche proposée dans [VD10] pour la conception de ce TRNG est assez pertinente. Dans un premier temps, les auteurs étudient et caractérisent la cellule TERO. Ils montrent que celle-ci est plus sensible au bruit que les oscillateurs en anneaux à inverseurs classiques. La comparaison est étudiée sur le plan mathématique et validée par des simulations LT Spice ainsi que des mesures dans une cible FPGA. Les auteurs proposent ensuite un modèle mathématique du

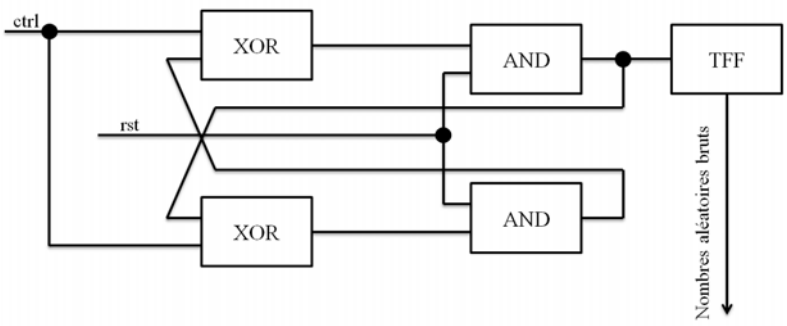


Figure 1.11 – Schéma d'une cellule de base du TRNG proposé dans [VD10]

TRNG qui, bien qu'il ne rendre pas compte des taux d'entropie en sortie (il ne s'agit pas d'un modèle stochastique), décrit bien les mécanismes de fonctionnement du générateur. Enfin, les auteurs proposent une implantation du TRNG dans une cible Xilinx Spartan 4. L'architecture sélectionnée comporte 2 cellules TERO, elle fourni des suites statistiques validées à l'aide des tests NIST SP 800-22 avec un débit de 250 kb/s sans post-traitement.

Annexe B

Conception des circuits asynchrones

Cette annexe vient en complément de la section 2.2 du chapitre 2. Elle décrit les protocoles de communication et les types de codage dans les circuits asynchrones. Ensuite elle présente les différentes classes de circuits asynchrones.

1. Protocoles de communication

Deux protocoles de communication sont couramment utilisés dans l'implantation de circuits asynchrones : le protocole 2 phases, dit NRZ (sans retour à zéro, *half-handshake*), et le protocole 4 phases, dit RZ (retour à zéro, *full-handshake*).

Le protocole 2 phases correspond à la séquence minimale nécessaire à l'implantation d'une communication de type requête/acquittement. Son principe, décrit dans Fig. 2.1, s'appuie sur deux phases de fonctionnement :

- *Phase 1* : détection de la requête de l'émetteur par le récepteur, traitement de la donnée puis génération d'un signal d'acquittement. C'est la phase active du récepteur.
- *Phase 2* : détection du signal d'acquittement par l'émetteur (signalant la consommation d'une donnée). Le canal de communication est alors prêt à recevoir une nouvelle donnée. Il s'agit de la phase active de l'émetteur.

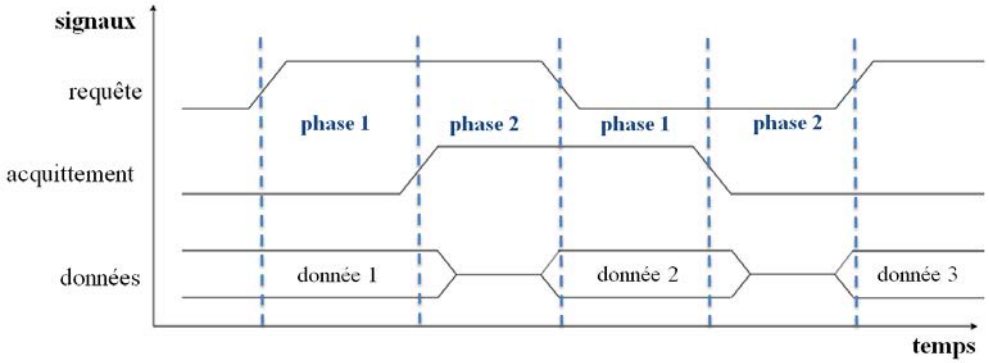


Figure 2.1 – Protocole de communication 2 phases

Le protocole 4 phases est décrit Fig. 2.2. Comme son nom l'indique, il s'appuie sur 4 phases de fonctionnement :

- *Phase 1* : détection de la requête de l'émetteur par le récepteur, traitement de la donnée et génération du signal d'acquittement.
- *Phase 2* : détection de l'acquittement par le récepteur, les données sont invalidées et la requête remise à zéro.
- *Phase 3* : détection de la remise à zéro par le récepteur et remise à zéro du signal d'acquittement.
- *Phase 4* : détection de la remise à zéro de l'acquittement par l'émetteur, celui-ci peut enclencher une nouvelle communication dès qu'une donnée est disponible à son entrée.

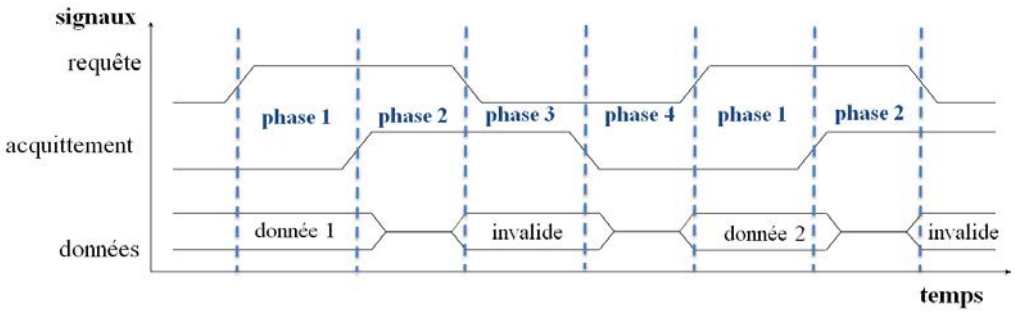


FIGURE 2.2 – Protocole de communication 4 phases

En terme de performances et de consommation, le protocole 2 phase peut sembler à priori plus efficace puisqu'il nécessite deux fois moins de transitions que le protocole 4 phases. Ceci est en réalité inexact. D'une part, le protocole 2 phases nécessite la détection de transitions et non pas de niveaux, ce qui impose l'utilisation de matériel plus complexe (et plus consommant). D'autre part, les techniques d'optimisation de pipelines asynchrones permettent souvent de s'affranchir de la pénalité apparente des phases de retour à zéro.

2. Codage des données

Tandis que dans la logique synchrone le codage des données est implicite, la logique asynchrone nécessite à la fois de coder les données mais aussi leur validité. Pour cela, deux solutions se distinguent :

- *Codage "données groupées"* : un signal de validité est explicitement ajouté en parallèle aux signaux de données. La valeur de la donnée est codée comme en logique synchrone, i.e. un rail par bit de donnée (*single rail*), et le signal de validité par un rail supplémentaire. L'inconvénient de ce type de codage est la sensibilité aux délais (l'occurrence de la génération du signal de requête est calibrée par rapport au chemin critique de l'opérateur).
- *Les codages insensibles au délais* : l'information de validité est encapsulée avec le signal de donnée. Un bit de donnée requiert ainsi deux rails dont la combinaison contient à la fois la

donnée et l'information de validité (*dual rail*). Deux codages sont communément adoptés : le codage trois états et le codage quatre états. Dans le codage 3 états, un rail est activé pour la valeur '1' et l'autre pour la valeur '0'. L'état "11" est interdit tandis que l'état "00" correspond à des données invalides. Ce codage est particulièrement adapté au protocole de communication 4 phases vu que le passage d'une valeur valide à une autre implique nécessairement le passage par l'état invalide. Dans le codage 4 états, la valeur de la donnée est codée dans la parité du nombre donné par les deux rails. Chaque fois qu'une donnée est émise, la parité est changée, ce qui permet de passer d'une donnée à une autre sans passer par un état invalide. Ce type de codage est donc particulièrement adapté au protocole de communication 2 phases.

3. Classification des circuits asynchrones

Les circuits asynchrones sont classifiés selon leurs hypothèses temporelles correspondant généralement aux modèles de délais utilisés pour les opérateurs et les fils d'inter-connexion. Plus ces hypothèses sont relachées, plus le circuit est simple et robuste. Fig. 2.3 représente ces différentes classes de circuits asynchrones en fonction de leurs hypothèses temporelles et robustesse.

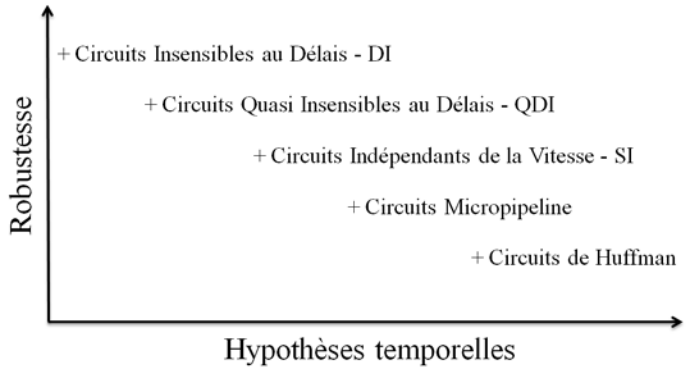


Figure 2.3 – Classification des circuits asynhcrones selon leurs hypothèses temporelles

Il existe cinq grandes classes de circuits asynchrones :

- *Circuits insensibles aux délais - DI (Delay Insensitive)* : le modèle de délai utilisé pour les portes et interconnexions est un modèle non-borné. Le fonctionnement de ce type de circuits est garanti quelque soient les délais introduits dans ses éléments. En pratique, ce type de conception est très restrictif car il nécessite l'utilisation de portes complexes à plusieurs entrées et plusieurs sorties. En effet, le changement d'état de la sortie de ce type de portes ne peut servir à acquitter qu'un seul des opérateurs connectés à ses entrées, ce qui exclue d'emblée l'utilisation de portes logiques classiques.
- *Circuits quasi-insensibles aux délais - QDI (Quasi Delay Insensitive)* : Le modèle utilisé pour les délais est également non borné mais une hypothèse temporelle est ajoutée. Cette

hypothèse, dite de fourches isochrones, suppose que deux fils qui relient la sortie d'un élément aux entrées de deux éléments différents (une fourche) ont le même temps de propagation. Ainsi, en émettant l'hypothèse que les fourches d'acquittement sont isochrones, il est possible d'utiliser le changement d'état de la sortie d'une porte logique standard pour acquitter tous les éléments connectés à ses entrées. L'implantation de cette classe de circuit est donc plus souple en pratique.

– *Circuits indépendants de la vitesse - SI (Speed Independent)* : Il s'agit de circuits QDI auxquels on rajoute l'hypothèse que toutes les fourches sont isochrones. Le modèle de délai est également supposé non borné et les temps de propagation dans les fils sont supposés nuls. Cette hypothèse simplifie la conception des circuits QDI dans l'optique où des outils spécifiques permettraient d'effectuer les vérifications nécessaires uniquement sur les fourches sensibles.
– *Circuits de type "Micropipeline"* : introduit par I. E. Sutherland, ce type de circuits permet de contrôler la propagation de données au sein d'une file de type FIFO grâce à des cellules de Muller (*First In, First Out*) comme décrit dans Fig. 2.4. Les registres utilisés sont d'un type particulier (appelés *"event-controlled storage elements"*), ils fonctionnent de la manière suivante : l'entrée *C* (capture) permet la capture de la donnée en entrée, la sortie *Cd* est activée lorsque la donnée est mémorisée. L'entrée *P* active le registre (laissant la donnée circuler de son entrée vers sa sortie) et la sortie *Pd* est quand elle activée quand le registre est passant (pass done). Les blocs combinatoires, quant à eux, peuvent être implantés de manière classique.

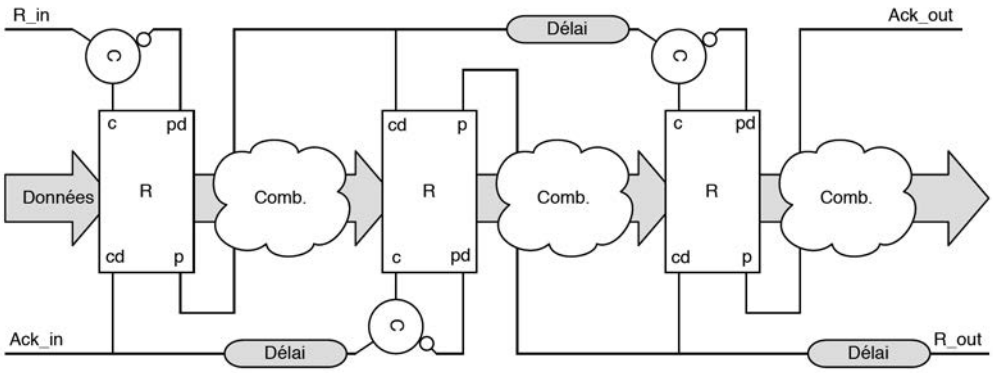


Figure 2.4 – Structure d'un circuit micropipeline (figure tirée de [Ham09])

Lorsqu'un événement survient sur *R_in* à l'entrée d'un étage du micropipeline, les données en entrée sont capturées dans le registre et maintenues à sa sortie pour permettre le traitement combinatoire. L'événement se propage alors vers *Ack_in* et arrive au deuxième étage après un délai calibré sur le temps d'exécution de ce premier bloc combinatoire. Si le registre de cet étage est dans l'état passant, alors les données en sortie du bloc combinatoire y sont capturées et le registre du premier étage est rendu passant à nouveau pour lui permettre de consommer une nouvelle donnée, et ainsi de suite.

Cette classe de circuits est moins robuste que les circuits QDI du fait de l'adoption d'un modèle de délai borné pour le chemin de données. Il est néanmoins possible de contourner cette limitation en utilisant des opérateurs combinatoires capables de générer leurs propres signaux de fin d'acquittement.

– *Circuits de Huffman* : Les modèles de délais sont bornés comme dans les circuits synchrones. La fiabilité de ce type de circuit dépend fortement des conditions de fonctionnement et s'appuie principalement sur une bonne caractérisation des temps de propagation dans les différents opérateurs. Il s'agit de loin de la classe de circuits asynchrones la moins robuste.

4. Propriétés et avantages des circuits asynchrones

Ce paragraphe présente brièvement les caractéristiques spécifiques aux circuits asynchrones ainsi que les avantages liés à leur utilisation.

• Absence d'horloge

Il s'agit de la propriété la plus significative des circuits asynchrones et dont découle plusieurs avantages de leur utilisation. L'absence d'horloge dans les circuits asynchrones permet de s'affranchir de beaucoup de problèmes et contraintes liées à sa présence. Ceci est de plus en plus vrai étant donné les développements récents des circuits numériques, avec en premier plan leur miniaturisation. Cette miniaturisation impose des contraintes de plus en plus importantes sur les circuits de distribution d'horloge qui deviennent prépondérants dans les systèmes synchrones en terme de complexité, surface et consommation. Leur conception peut vite devenir un facteur limitant les performances dans les circuits synchrones. *A contrario*, dans les circuits asynchrones, les éléments qui assurent le contrôle local des données sont distribués dans l'ensemble du circuit et sont donc plus faciles à maîtriser. Ceci élimine les pics de consommation liés au fronts d'horloge, la consommation globale du circuit est donc mieux répartie dans le temps ce qui limite considérablement le bruit dans les lignes d'alimentation.

De plus, pour certaines classes de circuits asynchrones, le fonctionnement est indépendant des délais introduits par les opérateurs, ce qui élimine d'emblée les problèmes liés au *clock skew* ainsi qu'à la gigue d'horloge (*jitter*). L'optimisation de la vitesse des circuits asynchrones repose donc uniquement sur l'optimisation des protocoles de communication et des cellules elle-mêmes.

• Calcul en temps minimal et robustesse aux conditions de fonctionnement

Les opérateurs asynchrones évaluent une fonction avec un temps variable qui dépend généralement des données en elles-même. De part les mécanismes de contrôle local cités précédemment, les données sont utilisables immédiatement après leur traitement tandis que, dans les circuits synchrones, le temps de traitement des données est limité par les temps les plus longs dans le circuit, ceux-ci peuvent en plus varier avec les conditions extérieures (température, tension d'alimentation) et la variabilité de procédé de fabrication. De ce fait, les circuits asynchrones sont potentiellement plus rapides et plus robustes que leurs homologues synchrones. Ce calcul en temps minimal a une autre conséquence sur les circuits de types "micropipeline". Dans une pipeline synchrone, le déplacement des données est provoqué par l'occurrence d'un front d'horloge. Les données sont donc séparées d'un nombre constant d'étage lorsqu'elle se propagent. Ceci n'est pas le cas dans une pi-

peline asynchrone qui est de ce fait "élastique", les données progressent aussi longtemps qu'elles ne rencontrent pas de ressources occupée indépendamment des données qui la suivent. Cette propriété est exploitée pour améliorer la vitesse et consommation des pipelines asynchrones.

• **Modularité et migration**

L'utilisation de mécanismes de contrôle local dans les circuits asynchrones fait qu'ils sont naturellement très modulaires. Les différents blocs d'un système peuvent être assemblés comme des briques de base sans se préoccuper des problèmes de synchronisation. Ceci permet aussi une distribution des tâches de manière plus aisée entre plusieurs concepteurs travaillant en parallèle sur un même système, ce qui favorise particulièrement la réutilisation de blocs et l'échange de propriétés intellectuelles (IP).

Le fonctionnement d'un circuit asynchrone est *a priori* indépendant de la réalisation des cellules qui le constituent pourvu que le protocole de communication soit respecté. Ainsi, il est facile de modifier l'implantation ou la technologie des cellules utilisées sans altérer l'aspect fonctionnel. Cette capacité de migration rend le style de conception asynchrone particulièrement attractif au vu de la rapidité des évolutions technologiques actuelles.

• **Faible consommation**

Cette propriété découle de plusieurs facteurs :

- *L'absence d'horloge* : dans les circuits récents, l'énergie dissipée par le système de distribution d'horloge et les éléments de mémorisation peut représenter jusqu'à 50% de la consommation d'un circuit.
- *La mise en veille* : un opérateur asynchrone qui ne traite pas de donnée ne consomme pas, les circuits asynchrones comprennent de manière intrinsèque une mise en veille locale des blocs qui ne fonctionnent pas à un moment donné.
- *L'absence d'aléas* : bien qu'ils ne sont pas gênants d'un point de vue fonctionnel dans les circuits synchrones, les aléas logiques impliquent quand même une consommation d'énergie supplémentaire inutile. La conception asynchrone suppose qu'il n'y ait aucun aléa afin d'obtenir des circuits corrects du point de vue fonctionnel.
- *Souplesse vis-à-vis des conditions de fonctionnement* : les circuits asynchrones étant plus robustes aux conditions de fonctionnement, il est aisé de réduire leur tension d'alimentation pour réduire leur consommation d'énergie.

• **Faible bruit, faible rayonnement électromagnétique**

Comme cité auparavant, le fonctionnement synchrone nécessite l'utilisation d'une horloge qui permet de séquencer les opérations. Celles-ci s'effectuent simultanément pendant un front d'horloge, ce qui induit de forts appels de courant dans les lignes d'alimentation à ces moments. Ceci provoque un rayonnement électromagnétique important concentré aux harmoniques de la fréquence d'horloge, mais également un bruit parasite déterministe sur les lignes d'alimentation et le substrat. Ces deux phénomènes peuvent être extrêmement gênants lorsque l'on manipule des signaux de très faible puissance. A contratrio, le rayonnement électromagnétique et le bruit sur les lignes d'alimentation est fortement limité de part l'absence d'horloge dans les circuits asynchrones (en réalité mieux répartis dans le temps). Cette propriété rend ces circuits particulièrement attractifs notamment pour les application de type RF (radio-fréquences).

• Circuits QDI pour les applications cryptographiques

A priori, les circuits asynchrones ne semblent pas particulièrement adaptés pour des applications cryptographique. Il s'avère en effet que l'analyse des courbes de courant des circuits synchrones est fortement brouillée par les effets parasites dus à la consommation des arbres d'horloge. Alors que dans les circuits asynchrones, l'absence d'horloge globale rend finalement l'analyse des courbes de consommation plus facile (celle-ci reflète directement l'activité des blocs qui fonctionnent à un moment donné).

Alors qu'il a été montré que les circuits de type micropipeline présentent des caractéristiques quasi semblables à celles des circuits synchrones face aux attaques en puissance de type DPA (Differential Power Analysis) [Bey09], il semble que les circuits de type QDI (quasi insensibles au délais) présentent une résistante accrue face aux attaques à canaux cachés [Bey09]. Cela découle de plusieurs propriétés :

- *Le contrôle local* : permet une répartition de l'activité non homogène dans le temps, ce qui augmente la difficulté de se synchroniser sur un événement particulier et rend donc les attaques temporelles moins efficaces.
- *Le protocole de communication 4 phases* : ce protocole suppose une remise à zéro des noeuds logiques entre deux calculs, tandis que dans la logique synchrone la consommation de courant dépend directement de la valeur précédente. Le protocole 4 phases permet donc une meilleure répartition de la consommation dans le temps et réduit sa dépendance aux données.
- *Les codages à 3 et 4 états* : permettent de garder un poids de Hamming constant quelles que soient les données manipulées ce qui réduit également la dépendance de la consommation aux données. Par ailleurs, l'état invalide peut servir à la génération d'alarmes en cas de détection de faute, ce qui offre un avantage supplémentaire contre les attaques par injection de faute.

Bibliographie

[BB99] V. Bagini and M. Bucci. A design of reliable true random number generator for cryptographic applications. In *1st Workshop on Cryptographic Hardware and Embedded Systems - CHES 1999*, volume 1717 of *LNCS*, pages 204–218. Worcester, USA, Springer Verlag, 1999.

[BBA+12] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. Contactless electromagnetic active attack on ring oscillator based true random number generator. In Werner Schindler and Sorin A. Huss, editors, *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 2012.

[BBAF13] Pierre Bayon, Lilian Bossuet, Alain Aubert, and Viktor Fischer. Electromagnetic analysis on ring oscillator-based true random number generators. In *ISCAS*, pages 1954–1957. IEEE, 2013.

[BBFV10] Nathalie Bochard, Florent Bernard, Viktor Fischer, and Boyan Valtchanov. True-randomness and pseudo-randomness in ring oscillator-based true random number generators. *Int. J. Reconfig. Comp.*, 2010, 2010.

[BBS86] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2) :364–383, 1986.

[Bey09] Taha Beyrouthi. *Logique programmable asynchrone pour systèmes embarqués sécurisés.* PhD thesis, Institut National Polytechnique de Grenoble (INPG), 2009.

[BFV10] Florent Bernard, Viktor Fischer, and Boyan Valtchanov. Mathematical model of physical rngs based on coherent sampling. *Tatra Mountains Mathematical Publications*, 45 :1–14, 2010.

[BGL+03] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, and M. Varanonuovo. A High-Speed Oscillator-Based Truly Random Number Source for Cryptographic Applications on a Smart Card IC. *IEEE TRANSACTIONS ON COMPUTERS*, pages 403–409, 2003.

[BGL+06] M. Bucci, L. Giancane, R. Luzzi, M. Varanonuovo, A. Trifiletti, I.T. AG, and A. Graz. A novel concept for stateless random bit generators in cryptographic applications. *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, page 4, 2006.

[BLMT11] Mathieu Baudet, David Lubicz, Julien Micolod, and André Tassiaux. On the security of oscillator-based random number generators. *Journal of Cryptology*, 24(2) :398–425, 2011.

[CFAF12] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. Comparison of self-timed ring and inverter ring oscillators as entropy sources in fpgas. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 1325–1330, March 2012.

[CFAF13] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. A self-timed ring based true random number generator. In *Asynchronous Circuits and Systems (ASYNC), 2013 IEEE 19th International Symposium on*, pages 99–106, May 2013.

[CFFA13] Abdelkarim Cherkaoui, Viktor Fischer, Laurent Fesquet, and Alain Aubert. A very high speed true random number generator with entropy assessment. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 179–196. Springer Berlin Heidelberg, 2013.

[Cla67] Wesley A. Clark. Macromodular computer systems. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 335–336, New York, NY, USA, 1967. ACM.

[CM73] T.J. Chaney and C.E. Molnar. Anomalous behavior of synchronizer and arbiter circuits. *Computers, IEEE Transactions on*, C-22(4) :421–422, April 1973.

[Cor66] Rand Corporation. *A million random digits with 100,000 normal deviates*. Free Press New York, 1966.

[Dav78] A. L. Davis. The architecture and system method of ddm1 : A recursively structured data driven machine. In *Proceedings of the 5th Annual Symposium on Computer Architecture*, ISCA '78, pages 210–215, New York, NY, USA, 1978. ACM.

[Dav02] R. B. Davies. Exclusive OR (XOR) and hardware random number generators. Online. Available at : http ://webnz.com/robert/, February 2002.

[DG07] M. Dichtl and J.D. Golic. High-Speed True Random Number Generation with Logic Gates Only. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *LNCS*, pages 45–61. Vienna, Austria, Springer Verlag, 2007.

[DGH07] J.L. Danger, S. Guilley, and P. Hoogvorst. Fast True Random Generator in FPGAs. *Circuits and Systems, 2007. NEWCAS 2007. IEEE Northeast Workshop on*, pages 506–509, 2007.

[DGH09] Jean-Luc Danger, Sylvain Guilley, and Philippe Hoogvorst. High speed true random number generator based on open loop structures in fpgas. *Microelectronics Journal*, 40(11) :1650–1656, 2009.

[Dic03] Markus Dichtl. How to predict the output of a hardware random number generator. In ColinD. Walter, ÇetinK. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 181–188. Springer Berlin Heidelberg, 2003.

[EFS98] Jo C. Ebergen, Scott Fairbanks, and Ivan E. Sutherland. Predicting performance of micropipelines using charlie diagrams. In *ASYNC*, pages 238–246. IEEE Computer Society, 1998.

[EHK+03] M. Epstein, L. Hars, R. Krasinski, M. Rosner, and H. Zheng. Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts. In *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *LNCS*, pages 152–165. Cologne, Germany, Springer Verlag, 2003.

[EYRF10] Oussama Elissati, Eslam Yahya, Sébastien Rieubon, and Laurent Fesquet. A high-speed high-resolution low-phase noise oscillator using self-timed rings. In *VLSI-SoC*, pages 173–178. IEEE, 2010.

[Fai09] S Fairbanks. High Precision Timing using Self-timed Circuits. Technical report, University of Cambridge, Computer Laboratory, url : http ://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-738.pdf, 2009.

[FBBV08] Viktor Fischer, Florent Bernard, Nathalie Bochard, and Michal Varchola. Enhancing security of ring oscillator-based trng implemented in fpga. In *FPL*, pages 245–250. IEEE, 2008.

[FBH13] V. Fischer, F. Bernard, and P. Haddad. An open-source multi-fpga modular system for fair benchmarking of true random number generators. In *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pages 1–4, Sept 2013.

[FCN13] V. Fischer, Zouha Cherif, and Xuan Ngo. A puf based on transient effect ring oscillator and insensitive to locking phenomenon. *IEEE Transactions on Emerging Topics in Computing*, 99(PrePrints) :1, 2013.

[FD02] Viktor Fischer and Milos Drutarovsky. True random number generator embedded in reconfigurable hardware. In *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002), Springer-Verlag, LNCS 2523*, pages 415–430. Springer-Verlag, 2002.

[FDSB04] V. Fischer, M. Drutarovsky, M. Simka, and N. Bochard. High performance True Random Number Generator in Altera Stratix FPLDs. *Lecture notes in computer science, FPL'04*, pages 555–564, 2004.

[Fis12] Viktor Fischer. A closer look at security in random number generators design. In Werner Schindler and SorinA. Huss, editors, *Constructive Side-Channel Analysis and Secure Design*, volume 7275 of *Lecture Notes in Computer Science*, pages 167–182. Springer Berlin Heidelberg, 2012.

[FM04] Scott Fairbanks and Simon W. Moore. Analog micropipeline rings for high precision timing. In *ASYNC*, pages 41–50, 2004.

[FMC85] R.C. Fairfield, R.L. Mortenson, and K.B. Coulthart. An lsi random number generator (rng). In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 203–230. Springer Berlin Heidelberg, 1985.

[FS03] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 2003.

[Gol04] J.D. Golic. New paradigms for digital generation and post-processing of random data. Technical report, Cryptology ePrint Archive, Report 2004/254, 2004.Online. Available : http ://eprint.iacr.org/2004/254.ps, 2004.

[Gol06] J.D. Golic. New Methods for Digital Generation and Postprocessing of Random Data. *IEEE TRANSACTIONS ON COMPUTERS*, pages 1217–1229, 2006.

[GWG02] Mark Greenstreet, Anthony Winstanley, and Aurelien Garivier. An event spacing experiment. In *Proceedings of the 8th International Symposium on Asynchronus Circuits and Systems*, ASYNC '02, pages 47–, Washington, DC, USA, 2002. IEEE Computer Society.

[Ham09] Jérémie Hamon. *Oscillateurs et architectures asynchrones pour le traitement des signaux radio impulsionnelle UWB.* PhD thesis, Institut National Polytechnique de Grenoble (INPG), 2009.

[HBF07] D. E. Holcomb, W. P. Burleson, and K. Fu. Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags. *In Proceedings of the Conference on RFID Security*, 2007.

[HFMR08] Jeremie Hamon, Laurent Fesquet, Benoit Miscopein, and Marc Renaudin. High-level time-accurate model for the design of self-timed ring oscillators. *2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems*, 0 :29–38, 2008.

[HFMR09] J. Hamon, L. Fesquet, B. Miscopein, and M. Renaudin. Constrained asynchronous ring structures for robust digital oscillators. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(7) :907–919, July 2009.

[Huf54] D. A. Huffman. The synthesis of sequential switching circuits. Technical report, Technical report, Research Laboratory of Electronics, Massachusetts Institute of Technology., 1954.

[Iss10] Oussama El Issati. *Les Oscillateurs Asynchrones en Anneau : de la Théorie à la Pratique.* PhD thesis, Université Joseph Fourier de Grenoble, 2010.

[JK99] B. Jun and P. Kocher. The Intel Random Number Generator. *Cryptography Research Inc. white paper, Apr*, 1999.

[KG04] P. Kohlbrenner and K. Gaj. An Embedded True Random Number Generator for FPGAs. *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pages 71–78, 2004.

[KL06] S.H.M. Kwok and E.Y. Lam. FPGA-based High-speed True Random Number Generator for Cryptographic Applications. *TENCON 2006. 2006 IEEE Region 10 Conference*, pages 1–4, 2006.

[KS01] W. Killmann and W. Schindler. AIS 31 : Functionality classes and evaluation methodology for true (physical) random number generators. *Bundesamt fur Sicherheit in der Informationstechnik (BSI), Bonn*, 2001.

[KS08] W. Killmann and W. Schindler. A Design for a Physical RNG with Robust Entropy Estimators. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware*

and Embedded Systems – CHES 2008, volume 5154 of *LNCS*, pages 146–163. Springer, 2008.

[KS11] W. Killmann and W. Schindler. AIS 31 : Functionality classes and evaluation methodology for true (physical) random number generators, version 2.0. *Bundesamt fur Sicherheit in der Informationstechnik (BSI), Bonn*, 2011.

[KSF99] John Kelsey, Bruce Schneier, and Niels Ferguson. Yarrow-160 : Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. In *In Sixth Annual Workshop on Selected Areas in Cryptography*, pages 13–33. Springer, 1999.

[Lab94] Information Technology Laboratory. Security requirements for cryptographic modules. Special Publication, January, 1994.

[Lab01] Information Technology Laboratory. Security requirements for cryptographic modules. Special Publication, May, 2001.

[Leh51] D. H. Lehmer. Mathematical methods in large-scale computing units. *Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery, 1949, pp. 141–146. Harvard University Press, Cambridge, Mass.*, 1951.

[Mau92] U.M. Maurer. A universal statistical test for random bit generators. *Journal of Cryptology*, 5(2) :89–105, 1992.

[MB556] A theory of asynchronous circuits i. Springer Berlin Heidelberg, 1956.

[MKD11] Mehrdad Majzoobi, Farinaz Koushanfar, and Srinivas Devadas. Fpga-based true random number generation using circuit metastability with adaptive feedback control. In *Proceedings of the 13th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'11, pages 17–32, Berlin, Heidelberg, 2011. Springer-Verlag.

[Ren10] M Renaudin. Etat de l'art sur la conception des circuits asynchrones : perspectives pour l'integration des systemes complexes. Technical report, Rapport technique, laboratoire TIMA„ 2010.

[RSN+01] A. Rukhin, J. Soto, J. Nechvatal, J. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications, nist special publication 800-22. Online. Available at : http ://csrc.nist.gov/, 2001.

[Sei70] Charles L. Seitz. Record of the project mac conference on concurrent systems and parallel computation. chapter Asynchronous Machines Exhibiting Concurrency, pages 93–106. ACM, New York, NY, USA, 1970.

[SMS07] B. Sunar, W.J. Martin, and D.R. Stinson. A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. *IEEE TRANSACTIONS ON COMPUTERS*, pages 109–119, 2007.

[SMTS13] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. Non-invasive spoofing attacks for anti-lock braking systems. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume

8086 of *Lecture Notes in Computer Science*, pages 55–72. Springer Berlin Heidelberg, 2013.

[SS05] Andrey Sidorenko and Berry Schoenmakers. Concrete security of the blum-blum-shub pseudorandom generator. In NigelP. Smart, editor, *Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 355–375. Springer Berlin Heidelberg, 2005.

[Sut89] I. E. Sutherland. Micropipelines. *Commun. ACM*, 32(6) :720–738, June 1989.

[SW12] Sergei Skorobogatov and Christopher Woods. Breakthrough silicon scanning discovers backdoor in military chip. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 23–40. Springer Berlin Heidelberg, 2012.

[Tka03] T.E. Tkacik. A Hardware Random Number Generator. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *LNCS*, pages 450–453. Redwood Shores, CA, USA, Springer Verlag, 2003.

[TKI+97] Akihiro Takamura, Masashi Kuwako, Masashi Imai, Taro Fujii, Motokazu Ozawa, Izumi Fukasaku, Yoichiro Ueno, and Takashi Nanya. Titac-2 : An asynchronous 32-bit microprocessor based on scalable-delay-insensitive model. In *ICCD*, pages 288–294, 1997.

[TLL03] K.H. Tsoi, K.H. Leung, and P.H.W. Leong. Compact FPGA-based true and pseudo random number generators. *Field-Programmable Custom Computing Machines, 2003. FCCM 2003. 11th Annual IEEE Symposium on*, pages 51–61, 2003.

[TLL07] K.H. Tsoi, K.H. Leung, and P.H.W. Leong. High performance physical random number generator. *IET Comput. Digit. Tech.*, 4(1) :349–352, January 2007.

[Val10] Boyan Valtchanov. *Générateurs de suites binaires vraiment aléatoires : modélisation et implantation dans des cibles FPGA*. PhD thesis, Université Jean Monnet de Saint-Etienne, 2010.

[VD10] Michal Varchola and Milos Drutarovsky. New high entropy element for fpga based true random number generators. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2010.

[VFA09] B. Valtchanov, V. Fischer, and A. Aubert. Enhanced trng based on the coherent sampling. In *Signals, Circuits and Systems (SCS), 2009 3rd International Conference on*, pages 1–6, Nov 2009.

[VHKK08] I. Vasyltsov, E. Hambardzumyan, Y.-S. Kim, and B. Karpinskyy. Fast Digital TRNG Based on Metastable Ring Oscillator. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *LNCS*, pages 164–180. Springer, 2008.

[VN51] John Von Neumann. Various Techniques Used in Connection with Random Digits. *J. Res. Nat. Bur. Stand.*, 12 :36–38, 1951.

[WG01] Anthony Winstanley and Mark R. Greenstreet. Temporal properties of self-timed rings. In Tiziana Margaria and Thomas F. Melham, editors, *CHARME*, volume 2144 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 2001.

[WT08] K. Wold and C. H. Tan. Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings. *2008 International Conference on Reconfigurable Computing and FPGAs*, pages 385 – 390, 2008.

[YEZ+09] E. Yahya, O. Elissati, H. Zakaria, L. Fesquet, and M. Renaudin. Programmable/stoppable oscillator based on self-timed rings. In *Asynchronous Circuits and Systems, 2009. ASYNC '09. 15th IEEE Symposium on*, pages 3–12, May 2009.

[YKBS10] Sang-Kyung Yoo, Deniz Karakoyunlu, Berk Birand, and Berk Sunar. Improving the robustness of ring oscillator trngs. *TRETS*, 3(2) :9, 2010.

Table des figures

Liste des tableaux

Table des matières